Beyond 5G Multi-Tenant Private Networks Integrating Cellular, Wi-Fi, and LiFi, Powered by Artificial Intelligence and Intent Based Policy

5G-CLARITY Deliverable D4.1

Initial Design of the SDN/NFV Platform and Identification of Target 5G-CLARITY ML Algorithms

Contractual Date of Delivery:	**October 31, 2020**
Actual Date of Delivery:	**October 31, 2020**
Editor(s):	**Daniel Camps-Mur, Hamzeh Khalili (I2CAT),**
Author(s):	**Erik Aumayr, Sven van der Meer (LMI),** **Pablo Ameigeiras, Jonathan Prados-Garzon, Oscar Adamuz-Hinojosa, Lorena Chinchilla, Pablo Muñoz (UGR),** **Alain Mourad, Ibrahim Hemadeh, Tezcan Cogalan (IDCC),** **Meysam Goodarzi, Jesús Gutiérrez, Vladica Sark, Najib Odhah (IHP),** **Rui Bian (PLF), Stefan Videv (USTRATH), Antonio Garcia (ACC),** **Carlos Colman Meixner, Shuangyi Yan, Xueqing Zou (UNIVBRIS),** **Jordi Pérez-Romero, Oriol Sallent, Irene Vilà, Ramon Ferrús (UPC),** **Jose Ordonez-Lucena (TID),** **Mir Ghoraishi (GIGASYS)**
Work Package:	**WP4**
Target Dissemination Level:	**Public**

This document has been produced in the course of 5G-CLARITY Project. The research leading to these results received funding from the European Commission H2020 Programme under grant agreement No. H2020-871428. All information in this document is provided “as is", there is no guarantee that the information is fit for any particular purpose. The user thereof uses the information at its own risk and liability. For the avoidance of all doubts, the European Commission has no liability in respect of this document, which is merely representing the authors view.

Revision History

Revision	Date	Editor /Commentator	Description of Edits
0.1	11.04.2020	Daniel Camps-Mur	High Level ToC Proposal
0.2	29.05.2020	Hamzeh Khalili	SOTA Section 2
0.3	03.06.2020	Hamzeh Khalili	Reviewing Section 2
0.4	30.07.2020	Hamzeh Khalili	Editing and reviewing section 2
0.5	08.09.2020	Alain Mourad, Ibrahim Hemadeh, Tezcan Cogalan, Jose Ordonez-Lucena	Merging Section 2.4.2 on telemetry initial design
0.55	15.09.2020	Daniel Camps-Mur, Hamzeh Khalili, Pablo Muñoz, Jose Ordonez-Lucena	Merging section 2.3.2 Slicing initial design
0.6	22.09.2020	Pablo Ameigeiras, Jonathan Prados-Garzon, Oscar Adamuz-Hinojosa, Lorena Chinchilla, Pablo Muñoz, Jordi Pérez-Romero, Oriol Sallent, Irene Vilà, Ramon Ferrús, Meysam Goodarzi, Jesus Gutierrez, Vladica Sark, Najib Odhah, Stefan Videv, Carlos Colman Meixner, Shuangyi Yan, Xueqing Zou, Alain Mourad, Ibrahim Hemadeh, Tezcan Cogalan	Merging ML algorithms in section 4
0.7	29.09.2020	Erik Aumayr	Merging section 5 on AI engine
0.8	06.10.2020	Jose Ordonez-Lucena, Pablo Muñoz	Merging section 3 on private public integration
0.9	13.10.2020	Sven van der Meer	Mergin Section 6 on Intent Engine
0.95	20.10.2020	Daniel Camps-Mur, Hamzeh Khalili	Added Section 1, executive summary, and conclusions
0.96	22.10.2020	Mir Ghoraishi (GIGASYS)	Full review and style correction
0.97	30.10.2020	Daniel Camps-Mur (I2CAT)	Addressing missing comment
1.0	31.10.2020	Jesús Gutiérrez (IHP) Mir Ghoraishi (GIGASYS)	Final review and 1.0 release

Table of Contents

List of Figures

List of Tables

List of Abbreviations

3GPP	3rd Generation Partnership Project
5GNR	5G New Radio
5G-ACIA	5G Alliance for Connected Industries and Automation
5GC	5G Core
ADR	Angle Diversity Receiver
ADT	Angle Diversity Transmitter
AF	Application Function
AFC	Automated Frequency Control
AGV	Automated Guided Vehicle
AI	Artificial Intelligence
AMF	Access and Mobility management Function
APD	Avalanche photodiode
API	Application Programming Interface
ATS	Asynchronous Traffic Shaper
AT3S	Access Traffic Splitting, Switching, Steering
AT3S-LL	AT3S Low Layer
B2B2X	Business-to-Business-to-X
B5G	Beyond 5G
BSI	Broadcast System Information
CaaS	Container as a Service
CAG	Closed Access Group
CBRS	Citizens Broadband Radio Service
CI/CD	Continuous Integration / Continuous Development
CNC	Central Network Controller
COTS	Commercial-Off-The-Shelf
CPE	Customer Premises Equipment
COP	Control Orchestration Protocol
CUC	Central User Controller
CUPS	Control User Plane Separation
DCSP	Data Centre Service Provider
DL AoD	Downlink Angle-of-Departure
DNN	Data Network Name
DPDK	Data Plane Development Kit
DSA	Dynamic Shared Access
E2E	End-to-End
EDCA	Enhanced Distributed Channel Access
eMBB	enhanced Mobile Broadband
eMBMS	Evolved Multimedia Broadcast Multicast Services
EPC	Evolved Packet Core
ETSI	European Telecommunications Standards Institute
FCC	Federal Communications Commission
FDD	Frequency Division Duplexing
FPGA	Field Programmable Gate Array

FQDN	Fully Qualified Domain Name
FWA	Fixed Wireless Access
gNB	next-generation Node B
gNB-CU	gNB Central Unit
gNB-DU	gNB Distributed Unit
gPTP	generic Precision Time Protocol
GSA	Global mobile Supplier Alliance
GSMA	Global System for Mobile Alliance
HCF	Hybrid coordination function
HLS	High-Layer Split
IaaS	Infrastructure as a Service
IAB	Integrated Access backhaul
ICT	Information and Communication Technologies
IETF	Internet Engineering Task Force
IWSN	Industrial Wireless Sensor Network
LGA	Land Grid Array
LLS	Low-Layer Split
LSA	License Shared Access
LWA	LTE WLAN Aggregation
LWIP	Level Integration with IPSEC Tunnel
MEC	Multi-access Edge Computing
MF	Management Function
ML	Machine Learning
MN	Master Node
mMTC	Massive Machine-Type Communications
MOCN	Multi-Operator Core Network
MPTCP	MultiPath Transmission Control Protocol
MR-DC	Multi-Radio Dual Connectivity
N3IWF	Non-3GPP InterWorking Function
near-RT	near-Real-Time
NEF	Network Exposure Function
NF	Network Function
NFV	Network Functions Virtualization
NFVI	NFV Infrastructure
NFVO	NFV Orchestrator
NG-RAN	Next Generation Radio Access Network
NGMN	Next Generation Mobile Networks
non-RT	non-Real-Time
NOP	Network Operator
NPN	Non-Public Network
NRM	Network Resource Model
NSaaS	Network Slice as a Service
NSD	Network Service Descriptor
NSMF	Network Slice Management Function
NSSAAF	Network Slice Specific Authentication Authorization Function

NSSAI	Network Slice Selection Assistance Identifier
NSSF	Network Slice Selection Function
NWDAF	NetWork Data Analytics Function
OBSS	Overlapping Basic Service Set
OFDM	Orthogonal Frequency Division Multiplexing
OOK	on-off-keying
OSS	Operation Support System(s)
OWC	Optical Wireless Communications
PAM	Pulse Amplitude Modulation
PCF	Policy Control Function
PDU	Packet Data Unit
PLMN	Public Land Mobile Network
PNF	Physical Network Function
PNI-NPN	Public Network Integrated NPN
PoC	Proof-of-Concept
PoT	Proof-of-Transit
PPM	Pulse Position Modulation
PRB	Physical Radio Block
QoS	Quality of Service
RGB	red, green and blue
RIC	RAN Intelligent Controller (RIC)
RRM	Radio Resource Management
RSS	Received Signal Strength
RU	Radio Unit
SAP	Service Access Point
SAS	Shared Access Spectrum
SBA	Service Based Architecture
SBMA	Service Based Management Architecture
SDN	Software Defined Networking
SDO	Standard Development Organization
SECF	Service Exposure Control Function
SLA	Service Level Agreement
SMF	Session Management Function
SN	Secondary Node
SNPN	Standalone NPN
SOA	Service Oriented Architecture
SON	Self-Organized Networks
SotA	State-of-the-Art
SPAD	Single-photon avalanche diodes
SRB	Signalling Radio Bearer
SSC	Session and Service Continuity
SotA	State-of-the-Art
SRP	Stream Reservation Protocol
TAS	Time-Aware Shaper
TAPI	Transport Application Programming Interface

TDD	Time Division Duplexing
TDMA	Time Division Medium Access
TNAN	Trusted non-3GPP access networks
TNGF	Trusted Non-3GPP Gateway Function
ToF	Time-of-Flight
TSC	Time Sensitive Communication
TSN	Time-Sensitive Networking
TTI	Transmission Time Interval
UDR	Unified Data Repository
UDSF	Unstructured Data Storage Function
UE	User Equipment
UICC	Universal Integrated Circuit Card
UNI	User Network Interface
UPF	User Plane Function
uRLLC	Ultra-Reliable Low Latency Communications
VAF	Virtualized Application Function
VCSEL	Vertical-Cavity Surface-Entity Laser
VDU	Virtual Deployment Unit
VIM	Virtualized Infrastructure Manager
VISP	Virtualized Infrastructure Service Provider
VLC	Visible Light Communication
VNF	Virtualized Network Function
VNFD	VNF Descriptor
VPN	Virtual Private Network
WAT	Wireless Access Technology
WDM	Wavelength Division Multiplexing
YAML	YAML (Yet Another Markup Language) Ain’t Markup Language
YANG	Yet Another Next Generation

Executive Summary

This document, 5G-CLARITY D4.1, is entitled “Initial design of the SDN/NFV platform and identification of target 5GCLARITY ML algorithms” and describes the work developed in WP4 “Management Plane” during the first year of the project.

The main contribution of this deliverable is a state-of-the-art analysis and an initial design for the main components of two stratums of the 5G-CLARITY architecture as defined in 5G-CLARITY D2.2, namely the **Management and Orchestration stratum** and the **Intelligence stratum**.

For the Management and Orchestration stratum, this deliverable defines in detail the Service and Slice management subsystem and the Data Processing subsystem:

- **Service and Slice Management subsystem**: A formal definition of a 5G-CLARITY slice and an initial design for the Slice Manager and the multi-WAT non real-time RAN Intelligent Controller are provided. A practical deployment of a 5G-CLARITY slice across the various wireless technologies, the transport network and the RAN and edge clusters is described.
- **Data Processing subsystem**: An initial design to gather multi-WAT telemetry in a RAN Intelligent Controller is provided. Two main components of this subsystem are identified and described, namely the Data Streaming Engine, which can gather and manipulate streaming data from multiple sources in the network, and the Data Lake that makes data available to the Machine Learning models living in the Intelligence Stratum.

For the Intelligence Stratum, this deliverable identifies Machine Learning use cases that are enabled by the 5G-CLARITY system and provides an initial design for the AI Engine and Intent Engine components:

- **ML models**: A set of 9 distinct Machine Learning models are identified that can be used to optimize network performance, including both non real-time and near real-time optimization.
- **AI Engine**: Is defined as a containerised execution environment that can manage the lifecycle of Machine Learning models. The main APIs between the AI Engine and other components of the 5G-CLARITY system are also identified.
- **Intent Engine**: A set of eight use cases are identified that illustrate the use of the Intent Engine within the 5G-CLARITY system, including interactions between the network operator and the Intent Engine and between individual Machine Learning models and the Intent Engine.

Overall, this document presents the main Management Plane innovations that will be developed in WP4 and sets the direction for the work that will be developed in 5G-CLARITY D4.2 and 5G-CLARITY D4.3.

1 Introduction

This deliverable, 5G-CLARITY D4.1, is the first report on the activities of the "WP4: Management Plane" in 5G-CLARITY. The deliverable is aligned with the 5G-CLARITY architectural principles presented in deliverable D2.2 and develops two of the introduced therein, namely the "Management and Orchestration stratum" and the "Intelligence stratum". The title of this deliverable is "Initial design of the SDN/NFV platform and identification of target 5G-CLARITY ML algorithms" and its main goal is to present the management plane innovations that will be developed in 5G-CLARITY throughput the duration of WP4, which include: i) design of slicing solutions for private networks, ii) design of integrated multi-WAT real-time telemetry systems, iii) novel mechanisms to integrate private and public networks, iv) network management algorithms powered by ML and hosted in an AI engine, and v) the design of intent based interfaces to ease the operation of private networks.

1.1 WP4 overview

5G-CLARITY WP4 works on the development of management systems for private networks. The relation of WP4 with other 5G-CLARITY Work Packages is depicted in Figure 1.1. The work in WP4 is driven by the use cases and architectural principles defined in WP2: "Scenario Descriptions, Architecture and Requirements", and is aligned to the work developed in WP3: "User and Control Plane", since the management systems defined in WP4 will be used to manage the functions developed in WP3. The output of WP4 is directly fed to WP5 "Integration, Experimentation, Proof-of-Concept and Demonstration", where it will be used both in self-contained demonstrations and in the project-wide use case demonstrations.

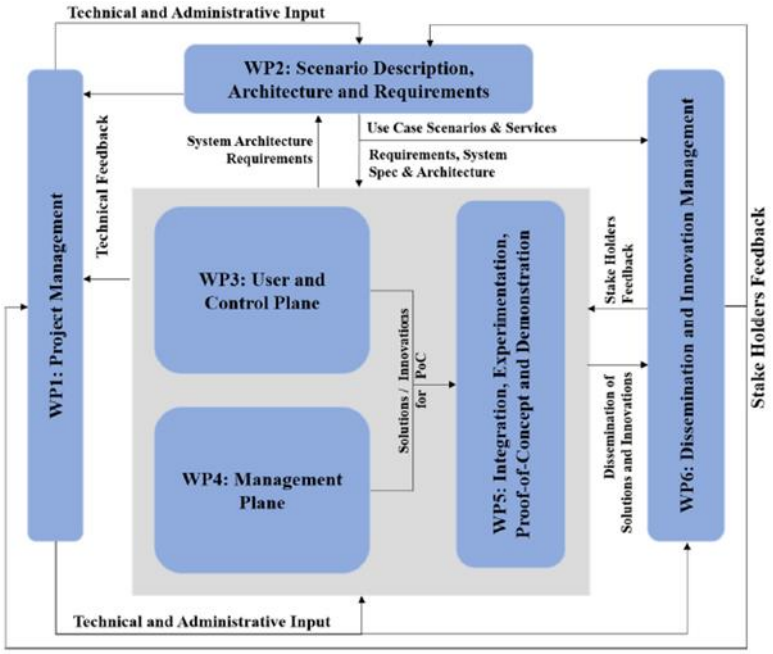


Figure 1.1 5G-CLARITY Work Package Structure.

The main objectives of WP4 are:

- O4.1. Design and demonstrate a multi-tenant SDN/NFV management platform allowing to configure the private 5G/Wi-Fi/LiFi network infrastructure and to provision third-party connectivity services in less than 5 minutes. (OBJ-TECH-6)
- O4.2. Design and demonstrate mechanisms to integrate the connectivity services provisioned over the 5G/Wi-Fi/LiFi infrastructure with an end-to-end (E2E) 5G slice in less than 10 minutes. (OBJ-TECH-

7)

- O4.3. Design and demonstrate an AI enabled engine, and the associated AI algorithms, which enables autonomic network management of the integrated 5G/Wi-Fi/LiFi infrastructure. (OBJ-TECH-8)

WP4 is articulated by means of 3 main tasks as follows:

- T4.1 Development of 5G/Wi-Fi/LiFi management platform, including policy language. The scope of this task is to design an SDN/NFV-based management platform to control the integrated 5G/Wi-Fi/LiFi, transport, and compute infrastructure available in the private venue. The planned platform will provide the following functionalities:
    - Allow the venue operator to provision separate infrastructure slices, including resource quotas to support connectivity services provisioned by third-party or vertical tenants.
    - Allow 3rd-party or vertical tenants to provision connectivity services for authorized devices to access the 5G/Wi-Fi/LiFi infrastructure.
    - Design an intent based policy language, acting as the platform's North Bound Interface (NBI), which can be used by the AI engine developed in T4.3 to manage the deployment and operation of the network.
- T4.2 Integration with E2E 5G slice framework. The scope of this task is to provide specifications on the architectural framework to be used for the management and orchestration of E2E network slices. The concept of E2E means that a network slice will allow subscribed UEs to connect to the 5G/Wi-Fi/LiFi APs within a private venue to reach MNO's service functions and applications – hosted in the Central Office (CO) outside the venue - with tailored QoS and with isolation guarantees. The referred framework must integrate the in-premises 5G/Wi-Fi/LiFi management platform with an enhanced MANO platform owned by the MNO.
- T4.3 AI engine development and learning algorithms, using historical network data for self-learning purpose. This task will support cognitive AI-driven processes that focus on both initial configurations to achieve high-level goals, and later "monitor and repair" activities. These processes will be guided by policies and data sources arising from T4.1 and will trigger/invoke management actions exposed by the E2E 5G slice management capabilities in T4.2. Appropriate ML-based analytics, policy-based decision making, and AI-based learning capabilities will be developed to allow initial intents to be achieved, and then support self-X capabilities to dynamically make decisions providing intelligent control mechanisms for achieving optimised performance, scalability flexibility and resilience.

This deliverable reports mostly on the work of T4.1 and T4.3, since these are the two tasks that have been active since the beginning of the project, while T4.2 will begin at month 13. However, discussions related to the integration of private and public networks, which are relevant to T4.2, have already happened in the project within the context of the architecture discussions in WP2, and also within the context of the slicing discussions in WP4. Therefore, Section 3 in this document is included to introduce the topics of private public integration that will be further developed in T4.2.

1.2 5G-CLARITY stratums requirements and initial designs

Since this deliverable presents the initial design for the 5G-CLARITY Management and Orchestration stratum and the Intelligence stratum, and for the sake of readability, a summary of the requirements and architecture designs for these two stratums are presented in this section derived from 5G-CLARITY D2.2 [2]. These initial solutions will be elaborated in the rest of this deliverable.

1.2.1 Management and orchestration stratum requirements and initial design

Table 1-1 contains the 5G-CLARITY Management and Orchestration stratum requirements introduced in 5G-CLARITY D2.2 [2].

Table 1-1 5G-CLARITY System Architecture Requirements.

Requirement ID	Requirement Description
CLARITY-MOS-R1	The 5G-CLARITY management and orchestration stratum shall be architected following the Service Based Management Architecture (SBMA) principles, with a set of MFs providing/consuming management services through a service bus.
CLARITY-MOS-R2	The 5G-CLARITY management and orchestration stratum shall allow for the provisioning of 5G-CLARITY resource-facing services (i.e. 5G-CLARITY wireless, compute and transport services).
CLARITY-MOS-R3	The 5G-CLARITY management and orchestration stratum shall keep a resource inventory, with information on the on-premise resources that can be used for the provision of 5G-CLARITY resource-facing services. This includes information on: *i)* the resource capacity of deployed wireless access nodes, including Wi-Fi/LiFi APs and physical gNBs; *ii)* the compute nodes available in the clustered NFVI (RAN cluster and edge cluster), and related computing/storage/networking resources; *ii)* the capacity and topology of deployed transport network.
CLARITY-MOS-R4	The 5G-CLARITY management and orchestration stratum shall store a catalog of VxFs/NSDs.
CLARITY-MOS-R5	The 5G-CLARITY management and orchestration stratum shall support to create, retrieve, update and delete VxFDs/NSDs
CLARITY-MOS-R6	The 5G-CLARITY management and orchestration stratum shall allow to create several instances of the same VxF/NFV service.
CLARITY-MOS-R7	The 5G-CLARITY management and orchestration stratum shall allow VxF / NFV service scaling. This scaling includes the scaling-in and scaling-out the resources of deployed VxF / NFV service instances.
CLARITY-MOS-R8	The 5G-CLARITY management and orchestration stratum shall allow for the provisioning of 5G-CLARITY slices, by defining separate resource quotas when allocating individual 5G-CLARITY resource-facing services.
CLARITY-MOS-R9	The 5G-CLARITY management and orchestration stratum shall maintain information regarding the mapping between 5G-CLARITY slices, constituent 5G-CLARITY resource-facing services and allocated resources.
CLARITY-MOS-R10	The 5G-CLARITY management and orchestration stratum shall allow resource elasticity and AI-assisted placement optimization as part of the 5G-CLARITY slice lifecycle management.
CLARITY-MOS-R11	The 5G-CLARITY management and orchestration stratum shall provide means for model-based data aggregation, with the ability to collect and process management data (e.g. performance measurements, fault alarms) from different sources in an automated and scalable manner.
CLARITY-MOS-R12	The 5G-CLARITY management and orchestration stratum shall be able to correlate aggregated data with deployed 5G-CLARITY slices and services instances, providing input to the intelligence engine for AI assisted operation of these instances.
CLARITY-MOS-R13	The 5G-CLARITY management and orchestration stratum shall provide necessary cloud-native capabilities for MF service production/consumption across the entire stratum.
CLARITY-MOS-R14	The 5G-CLARITY management and orchestration stratum shall allow individual 5G-CLARITY customers (e.g. MNOs) to securely access and consume MF services.
CLARITY-MOS-R15	The 5G-CLARITY management and orchestration stratum shall provide the means to expose capabilities with appropriate abstraction levels to individual 5G-CLARITY customers
CLARITY-MOS-R16	The 5G-CLARITY management and orchestration stratum shall provide isolation among customers' workflows and request

Figure 1.2 5G-CLARITY management and orchestration stratum: an SBMA approach., introduced in 5G-CLARITY D2.2, provides and overview of the Management Functions (MF) included in the Management and Orchestration stratum, which offer services to each other through a common Service Bus.

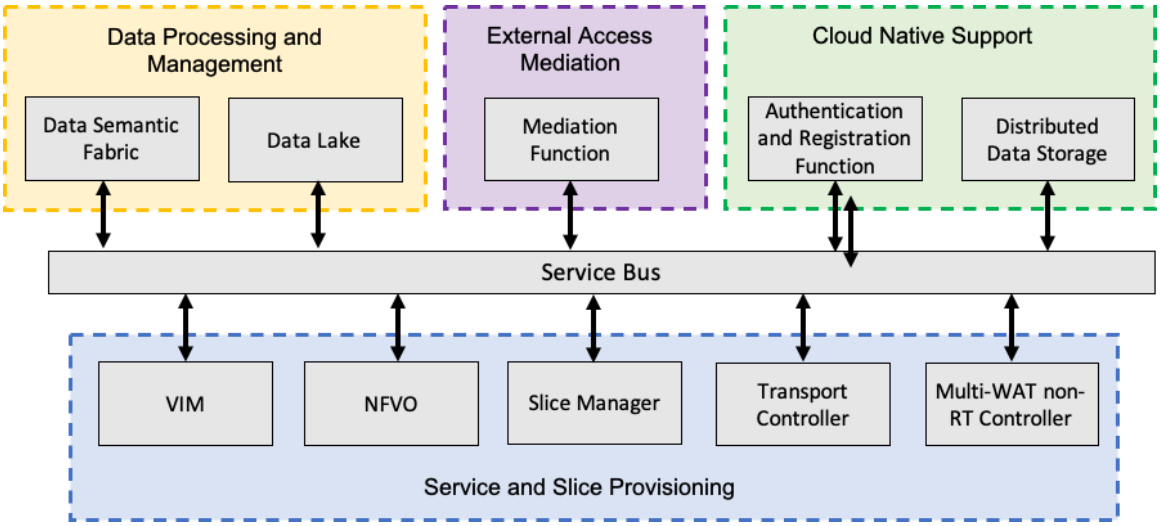


Figure 1.2 5G-CLARITY management and orchestration stratum: an SBMA approach.

The MFs composing the Management and Orchestration stratum can be arranged into four main functional groups:

- **Service and slice provisioning**: Grouping all the MFs that are involved in the provisioning of 5G-CLARITY slices, composed of wireless, transport and compute resources, as they have been defined in Section 2.3.2.
- **Data processing and management**: Grouping all the MFs that collect telemetry data from the physical and the Network Function stratums, in order to make the data available to the Intelligence stratum or to the external entities interacting with the management plane.
- **Cloud native support:** These are MFs required to enable a cloud native deployment of the 5G-CLARITY management and orchestration stratum. For example, enabling the deployment of stateless MFs that can be scaled dynamically in a cloud environment.
- **External access mediation**: Including MFs that police the interactions with external entities, such as the intelligence stratum or an external OSS, e.g. belonging to a mobile network operator (MNO).

A state-of-the-art (SotA) analysis and an initial design for the components of the Management and Orchestration stratum will be provided in this deliverable in Section 2 focusing on the two sub-components that require most innovations in the context of 5G-CLARITY, namely the Service and Slice Provisioning sub-system and the Data Processing and Management subsystem.

1.2.2 Intelligence stratum requirements and initial design

Table 1-2 contains the 5G-CLARITY Intelligence stratum requirements introduced in 5G-CLARITY D2.2 [2].

Table 1-2 Requirements for the 5G-CLARITY intelligence stratum.

Requirement ID	Description
CLARITY-INTS-R1	The 5G-CLARITY intelligence stratum shall leverage machine learning (ML) models to support intelligent management of network functions.

CLARITY-INTS-R2	The 5G-CLARITY intelligence stratum shall host ML models and offer them as services that are accessible outside of the intelligence stratum. Consumers of the ML services are either the network operator or other network functions.
CLARITY-INTS-R3	The 5G-CLARITY intelligence stratum shall provide a point of access for ML services to consume data from the network and forward recommended configurations to suitable network functions.
CLARITY-INTS-R4	The 5G-CLARITY intelligence stratum shall provide ML designers a process or interface to manage the lifecycle of ML models, including the deployment as services.
CLARITY-INTS-R5	The 5G-CLARITY intelligence stratum shall expose a communication interface towards the end user that simplifies the management of the 5G-CLARITY platform using intents, including intent-based network configuration and intent-based usage of available ML services.
CLARITY-INTS-R6	The 5G-CLARITY intelligence stratum shall expose an intent management interface through which the intent lifecycle can be controlled, including creation and removal.

Figure 1.3, introduced in D2.2, describes the two main components of the Intelligence stratum, namely the AI engine and the Intent engine, where the AI engine provides hosting and management of ML services, as well as aiding ML designers, and the Intent engine provides point of contact to and from the AI engine as well as a layer of abstraction towards the consumer of the AI functionalities, such as the network operator. Also shown are the high-level interfaces towards the operator, the ML designer, 5G-CLARITY network components (such as Slice Manager) and 5G-CLARITY Data Management.

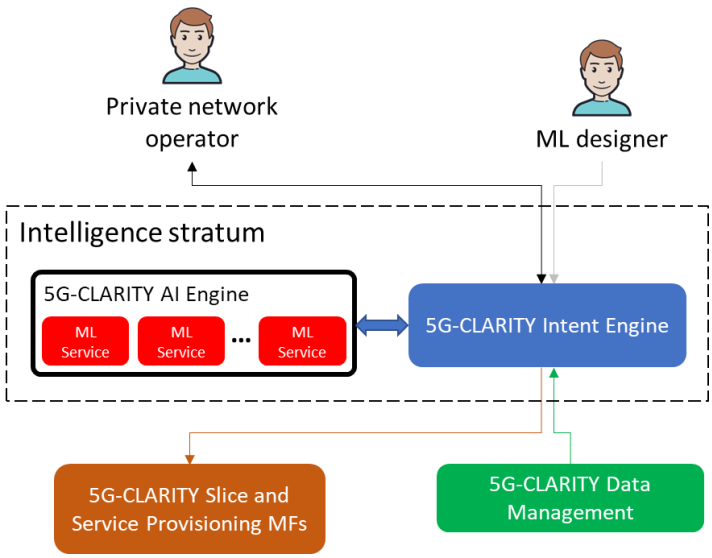


Figure 1.3 Architectural overview of the 5G-CLARITY intelligence stratum with its two main components, the AI engine and the Intent engine.

A state-of-the-art (SotA) analysis and an initial design for the components of the Intelligence stratum is provided in this deliverable in Section 4 (ML algorithms running in the AI engine), Section 5 (AI engine design) and Section 6 (Intent engine design).

1.3 Objective and scope of this document

Deliverable D4.1 takes input from 5G-CLARITY D2.1 [1], and has been co-developed alongside deliverable D2.2 [2], where WP4 has contributed some of the architectural aspects described in D2.2, which serve as basis for the initial designs presented in this document. This deliverable is also influenced by the user and

control functions reported in D3.1 [3], especially in the design of the ML algorithms in Section 5. The output of this deliverable will be used within WP4 to further develop the initial designs reported here towards D4.2. This deliverable will also be used in WP5 to define demonstration scenarios. This deliverable marks de acceptance criteria for milestone MS4.1. "Specification of Management Platform in North Bound Interface with AI engine and third-party MANO services".

D4.1 reports on state-of-the-art research and the definition of the initial innovations that will be executed in the various tasks of WP4, in the following way:

- T4.1: defining a new concept of network slice tailored to multi-tenant private network deployments, defining multi-WAT telemetry solutions, and defining and intent engine to ease the operation and configuration of the private network.
- T4.3: Introducing the concept of an AI engine used to host ML algorithms that will perform a set of network management functions and defining 9 ML algorithms that will be developed and demonstrated throughput the project.
- T4.2: Introducing an initial set of slice management models, explaining how 3rd parties (e.g. MNOs) can interact with the private network slices enabled by the systems defined in T4.1. Note that the actual work in T4.2 starts at month 13.

D4.1 has been carefully aligned with the architectural principles described in D2.2, and in particular this deliverable develops through its various chapters the "Management and Orchestration stratum" and the "Intelligence stratum" introduced in D2.2.

1.4 Document structure

The rest of this document is structured as follows:

- Chapter 2: Provides state-of-the-art research and initial solution design for the 5G-CLARITY network slicing subsystem for private networks, and for the 5G-CLARITY telemetry subsystem.
- Chapter 3: Provides an initial discussion on the interaction between 5G-CLARITY private networks and MNOs, which will be further developed in D4.2.
- Chapter 4: Describes a set of 9 ML algorithms that will be developed and demonstrated during the project.
- Chapter 5: Provides state-of-the-art research and initial solution design for the AI engine included in the 5G-CLARITY "Intelligence stratum" that is used to host and manage ML functions.
- Chapter 6: Provides state-of-the-art research and initial solution design for the Intent engine included in the 5G-CLARITY "Intelligence stratum".
- Chapter 7: Summarizes and concludes this document.

2 Management Platforms for Private Networks

2.1 Virtualization of Compute and Transport infrastructure in private networks

This section aims to provide an overview on the virtualization of cloud computing and transport technologies within the scope of the 5G-CLARITY ecosystem. For this, the most popular open source computing platform solutions and Software Defined Networking (SDN) controller technologies are listed.

2.1.1 State of the art

2.1.1.1 Virtualization on compute infrastructures

Cloud computing is the delivery of on demand computing services including servers, storage, databases, networking, applications and processing power over the Internet (on a pay as you go basis) to offer faster innovation, flexible resources, and economies of scale. Cloud computing platforms are integrated tools that provide management of cloud environments. These tools incorporate self-service interfaces, provisioning of system images, enabling metering and billing, and providing some degree of workload optimisation through established policies. This enables user to issue a request through a Cloud Controller to provision a virtual infrastructure somewhere on available resources within a Data Centre (DC). The Cloud Controller provides the central management system for cloud deployments. The most popular open source computing platforms are OpenStack and Kubernetes.

OpenStack [4] is a cloud operating system that offers an open source cloud computing platform for infrastructure as a service (IaaS) for both public and private clouds, where virtual servers and other resources are made available to users. OpenStack controls large pools of compute, storage, and networking resources in a DC, where all of them are managed and provisioned through APIs with common authentication mechanisms. These components are accessible through a unique dashboard that gives administrators complete control while empowering the end users to provision resources through a web interface. All OpenStack source code is available under an Apache 2.0 license.

OpenStack has a modular design that enables integration with legacy and third-party technologies. It is built on a shared-nothing, message-based architecture with modular components, each of which manages a different service. These services, together, instantiate an IaaS Cloud. OpenStack is composed of 10 key components. The primary component is the Nova compute service, which allows the end user to create and manage large number of virtual servers using the machine images. Nova is based on a modular architectural design where services can reside on a single host or, more commonly, on multiple hosts. OpenStack provisions and manages large networks of Virtual Machines (VMs) through open source libraries such as libvirt [5], where each VM requires its own underlying Operating System (OS) to run over a virtualized resource. The main disadvantage of a VM is that it can take up a lot of system resources. Each VM runs not just a full copy of an operating system, but a virtual copy of all the hardware that the operating system needs to run. This quickly adds up to a lot of RAM and CPU cycles. That is still economical better when compared to running separate actual computers.

Kubernetes [6] is a portable, extensible, open-source platform for managing containerized workloads and services, which empowers organizations to build and run scalable applications in modern, dynamic environments (i.e. public, private, and hybrid clouds). It works with a range of container tools, including Docker. Kubernetes defines a set of building blocks (primitives), which collectively provide mechanisms that deploy, maintain, and scale applications based on CPU, memory or custom metrics. Many cloud services offer a Kubernetes-based platform or infrastructure as a service (PaaS or IaaS) on which Kubernetes can be deployed as a platform-providing service.

Kubernetes includes services such as service architectures, infrastructure as a code, automation, continuous integration/delivery (CI/CD) pipelines, observability/monitoring tools, etc. Instead of VMs that requires its own underlying OS to run over a virtualized resources, Kubernetes is used to develop applications built with services packaged in containers, deployed as microservices and managed on elastic infrastructure through agile DevOps processes and continuous delivery workflows. Each container shares the host OS kernel and, usually, the binaries and libraries. This makes containers lighter weight than VMs where every instance has its own OS, binaries and libraries. In addition, containers only use the hardware resources that are needed at run- time, so there is no reservation of resources as in the case of VMs. Kubernetes is used as an industry de facto standard container orchestrator, which can be deployed either on the bare metal servers or on top of some virtualization technology.

Public Cloud is a platform that uses the standard cloud computing model to make resources (e.g. virtual machines (VMs), applications or storage) available to users remotely. Public cloud allows for scalability and resource sharing, which is not possible for a single organization to achieve. Public cloud architecture is categorized by the service model, including:

- Software as a service (SaaS), is a cloud model in which a third-party provider hosts application and makes them available to customers over the internet.
- Platform as a service (PaaS), is a computing model in which a third-party provider delivers hardware and software tools to its users as a service. So, it allows an organization to develop software without needing to maintain the underlying infrastructure.
- Infrastructure as a service (IaaS), in which a third-party provider offers virtualized computing resources, such as VMs and storage, over the internet or through dedicated connections. So, an organization outsources their entire DC to a cloud service provider. This model makes cloud adoption simpler.

The main examples of Public Clouds are providers such as Amazon Web Services, Google Cloud Platform and Microsoft Azure, which offer pay-per-usage deals that allow organizations to pay only for the resources they use.

Private Cloud is a cloud service that delivers similar advantages to public cloud, including scalability and self-service, but through a private infrastructure. Unlike public clouds that provide service to multiple organizations, the private cloud is a single tenant environment, which doesn't not share resources with any other organization. Resources can be hosted and managed in a variety of ways. Resources may be based on infrastructure already present in an organization's on-premise DC (managed and maintained by the organization internally) or on a separate infrastructure, which is provided by a third-party organization. Figure 2.1 shows the difference between public cloud and private cloud.

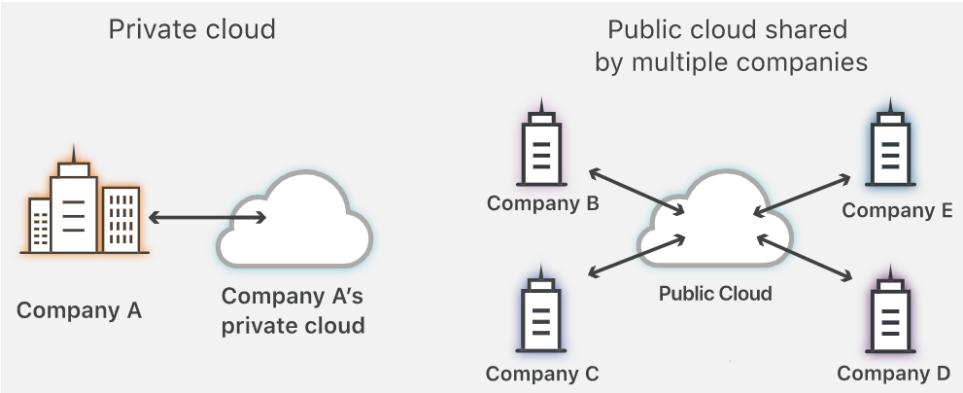


Figure 2.1 Comparison of Private and Public cloud.

2.1.1.2 SDN control of Ethernet transport technologies

SDN is a network architecture that enables network programmability by means of decoupling the Control Plane (CP) from the User Plane (UP), and by means of using open interfaces. As shown in Figure 2.2, the SDN architecture consists of three functional layers: infrastructure, control, and application. The infrastructure layer comprises a set of simple and low-cost network devices that can be configured and monitored through the southbound interface (SBI) by a logically centralized SDN controller (SDNC). The SDNC provides an abstraction of the network to the application layer. In this way, the SDN applications might consume network services and capabilities without being tied to their implementation details. The northbound interface (NBI) allows the interaction between the SDN applications and the SDNC. The SDN applications (e.g. telemetry, routing, firewalling, load balancing, policy enforcement, etc.) implement network intelligence. The SDN paradigm brings substantial benefits to network operators such as cost reduction, faster rollout of new services, and more granular service policies, among others. 5G-CLARITY considers Ethernet as the most relevant technology for providing enterprise networking services inside 5G private networks. Thus, we will focus on giving an overview of the SDN solutions for Ethernet-based transport technologies. More precisely, we will list the most representative existing frameworks and protocols for every SDN architectural component.

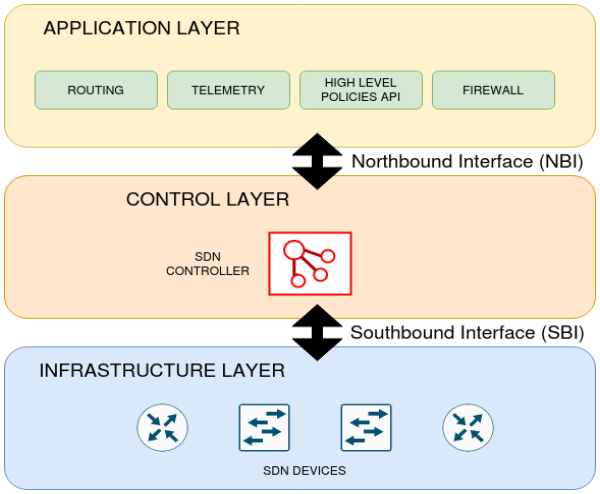


Figure 2.2 Basic SDN architecture proposed by the Open Networking Foundation [210].

Table 2-1: Description of the Most Representative SDN SBIs Solutions.

SBI Solution	Description
OF (OpenFlow) [7]	**Goal**: Complete SBI to enable SDNC to control and monitor the forwarding plane devices. OF was defined by the Open Networking Foundation (ONF) and is considered the first SDN standard and de facto standard in SDN. **Principle**: Set of well-defined messages that enables an SDNC to add, remove, or update flow entries from the OF compliant switches.
HAL (Hardware Abstraction Layer) [8]	**Goal**: Solve OF compatibility issues in legacy network elements. **Principle**: Set of abstractions. Two sublayers: *i)* the Cross-Hardware Platform Layer (CHPL), and *ii)* the Hardware-Specific Layer (HSL). CHPL is in charge of node abstraction, virtualization and configuration processes. HSL discovers the specific hardware platform and performs the required specific configurations.
POF (Protocol Oblivious Forwarding) [9]	**Goal**: Protocol-agnostic forwarding devices and data-path enhancement with stateful instructions. **Principle**: To use a generic flow instruction set (FIS)

OpenState [10]	**Goal**: Reduce OF signalling overhead and SDNC processing load. **Principle**: To leverage the Extended Finite State Machine (XFSM) to implement different stateful tasks inside the network devices (e.g. port knocking and MAC learning).
P4 [11]	**Goal**: Make OF more generic (protocol-agnostic) **Principle**: Flexible mechanism to parse packets and match header fields. It includes a programmable parser to define new headers. Unlike OF, it supports the parallelization of the Match and Action stages.
PAD (Programming Abstraction Datapath) [12]	**Goal**: Expose capabilities of network processors through a simple DP forwarding model valid for different types of networking hardware (e.g., programmable network processors, CPUs, optical devices, GEPON, DOCSIS). **Principle**: Creation of a library-based interface for managing and controlling DP protocols and DP functions.
DevoFlow [13]	**Goal**: Reduce OF signalling overhead **Principle**: The SDNC is only in charge of targeted and significant flows. The provision of aggregated flow statistics (this requires significant changes in the switch –costly-).
OvSDB (Open vSwitch Database Management Protocol) - RFC 7047- [14]	**Goal**: Complement OF with management and configuration capabilities out of OF scope. **Principle**: Management and configuration interface for long timescale operations (e.g., C/M/D of OF data-paths, configuration of a set of SDNCs, configuration of a set of managers, C/M/D of ports) in Open vSwitch.
OF-CONFIG (ONF-TS-016) [15]	**Goal**: Companion protocol to OF for configuring the OF operational context. **Principle**: Management and configuration interface for long timescale operations (e.g., OF controllers to OF DPs assignment, switch port enabling/disabling).
ForCES (Forwarding and Control Element Separation) [16]	**Goal**: SDN framework and SBI defined by IETF as an alternative to OF. **Principle**: A Network Element (NE) consists of Control Elements (CEs) and Forwarding Elements (FEs). A CE controls an NE via the ForCES protocol. This protocol also allows CEs to control and manage any ForCES-modelled FE. The FE abstraction models can be flexibly defined by the developers using the modelling language ForCES model.
OpFlex [17]	**Goal**: Proprietary SBI defined by Cisco through an IETF draft. **Principle**: OpFlex is based on a declarative policy information model, i.e., policy and management are centralized, whereas intelligence and control are distributed. There is a centralized Policy Repository (PR) that contains the policies and communicates with policy elements via OpFlex protocol. However, OpFlex does not include the functionality of programming the network from a centralized controller.
NetConf (Network Configuration Protocol) -RFC 6241- [18]	**Goal**: SBI intended to provide to provide a programmatic interface to the device that closely follows the functionality of the device's native interface. **Principle**: NetConf employs Remote Procedure Call (RPC) paradigm to realize the operations. It is a simple protocol for installing, manipulating, and deleting the configuration of network devices. It employs an Extensible Markup Language (XML)-based data encoding for the configuration data and the protocol messages.
SRv6 (Segment Routing IPv6) [19]	**Goal**: To provide a flexible and scalable mechanism for realizing source routing, i.e., the routing decisions are taken at the source and encoded in the packet header. **Principle**: Segment routing enables the addition of state information (a list of Segments) to packet headers. A Segment might instruct the packet steering over a given path or the packet delivery to a given service. This feature reduces the overhead at network devices and simplify and accelerate the service setup.

Among the SBIs listed in Table 2-1, 5G-CLARITY will focus on OF, OvSDB and NetConf to manage wireless and transport devices inside private venues.

In contrast to SBI, there has been little standardization effort for the NBI definition. The reason behind this might be the heterogeneity needs of the control and management applications and their dependence on the specific scenario. Despite this, most of the SDNCs include support for RESTful APIs, which employ HTTP requests, as NBI. Table 2-2 includes a comparison of commonly used open-source SDNCs, where their supported NBIs are listed. It shall be mentioned that these open-source controllers are used as a basis of many commercial SDN solutions provided by telecom equipment vendors.

Table 2-2: Qualitative Comparison of Popular Open Source SDNCs.

Controller	Characteristics
ONOS (Open Network Operating System) [20]	**Description:** Leading open source SDNC controller to build carrier-grade solutions targeted for service provider networks. **Features:** Web-based GUI, modular, easily extensible, horizontally scalable, telemetry support through pluggable modules, and resiliency through replication. **Supported SBIs**: OpenFlow, P4, NetConf, TL1, SNMP, BGP, RESTCONF, OvSDB, and PCEP. **Supported NBIs**: gRPC and RESTful APIs. **Programming Language**: Written in Java. **Community**: Linux Foundation Networking
ODL (OpenDayLight) [21]	**Description:** Open platform for customizing and automating networks of any size and scale. It was designed from the outset as a foundation for commercial solutions addressing wide range of use cases. It is suitable for SD-LAN and Cloud Integration spaces. **Features:** Web-based GUI, Modular, easily extensible, model-based approach (it requires a global, in-memory view of the network for logic computations), limited telemetry functionality, and resiliency through replication. **Supported SBIs**: OpenFlow, P4, NetConf, SNMP, BGP, RESTCONF, OvSDB, and PCEP. **Supported NBIs**: gRPC and RESTful APIs. **Programming Language**: Written in Java **Community**: Linux Foundation Networking. This project has the largest community support of all open source SDN controllers.
OpenKilda [22]	**Description:** Highly scalable SDNC architected from the ground up from web-scale technologies. It was devised to manage unreliable CP spanning across multiple carriers over long distances. It is successful in a distribution production environment. It combines Floodlight (SDNC), Kafka (message bus for telemetry), Apache Storm (storm-based cluster of agents for processing), OpenTSDB (data storage and analysis). **Features:** GUI, Modular, easily extensible, highly scalable, native support for telemetry, and resiliency through replication. **Supported SBIs**: OpenFlow **Supported NBIs**: RESTful APIs. **Programming Language**: Written in Java. **Community**: Much of the development and maintenance burden to its current users.
Ryu [23]	**Description:** Ryu is a component-based defined networking framework. It provides software components with well-defined API. Ryu can be regarded as a toolbox, with which SDNC functionality can be built. **Features:** Modular, easily extensible, no built-in clustering capability, no built-in telemetry functionality. **Supported SBIs**: OpenFlow, NetConf, OF-Config, and partial support of P4 **Supported NBIs**: RESTful APIs. **Programming Language**: Written in Python **Community**: Active community developing the framework, it is well supported.
Faucet [24]	**Description:** Compact OF SDNC, which enables network operators to run their networks the same way they do server clusters. Faucet moves network control functions to standard server-based software where those functions are readily manageable and extensible with

	common systems management approaches. It is built on top of Ryu, it includes two SB connections to the DP. One for configuration updates, the other (Gauge) for collecting and transmitting state information. **Features:** Modular, easily extensible, lightweight and highly scalable, built-in telemetry functionality, no built-in mechanisms for availability. **Supported SBIs**: OpenFlow 1.3 and support for feature such as VLANs, IPv4, IPv6, static, and BGP routing, port mirroring, policy-based forwarding and ACLs matching. **Supported NBIs**: YAML configuration files to track the target system state. It opens the SDN to administration by well-understood CI/CD pipelines and testing apparatus. **Programming Language**: Written in Python. **Community**: Active community developing the framework and it is well supported.

Regarding commercial SDN controllers, there are plenty of solutions out there. As a general rule, every vendor has its own solution. Many of these solutions are built on top of open source SDN controllers. Table 2-3 includes the primary functionalities of three commercial SDNCs from top telecom equipment vendors.

Table 2-3 Qualitative Comparison of Some Popular Commercial SDNCs.

Vendor/Controller or SDN Solution	Product Description
Ericsson / Ericsson Cloud SDN [25]	Network virtualization solution that provides connectivity for virtual, physical and container-based workloads. It combines an industrialized OpenDayLight controller with advanced routing capabilities. Key features: • Open: It is aligned with ETSI architecture and is based on open-source software (e.g., OpenDayLight and OVS). Integrated with OpenStack via standard APIs. • Enables Network Automation and Layer 3 services. • Developed and hardened for telecom grade deployments.
Huawei / Huawei Cloud SDN Solution (Agile Controller) [26]	Intended to large-scale data center networks. For NBIs, it uses RESTful APIs. For SBIs, it uses OpenFlow, OVSDB, and NetConf. Key features: • Automatic deployment. • Refined O&M. • Highly Reliable Clusters. • Open architecture. • Service-based network model customization. • Easy-to-use GUI for management. • Enables the rollout of new services within minutes. • Easy integration with mainstream cloud platforms.
NEC / PF6800 Programmable Flow Controller [27]	Three operation modes: *i)* OpenFlow Switch Fabric (OSF) which is an OF-based virtual network solution for switch control; *ii)* edge automation that employs traditional protocols for switch control; and *iii)* Hybrid mode that combines OSF and edge automation. Key features: • Easy-to-use management interface and drag-and-drop configuration. • Optimized and automated path selection with zero packet loss. • Telemmetry (accesible via a GUI).

Among the existing open-source and commercial SDN controllers 5G-CLARITY will favor ONOS and OpenDayLight that are the most widely adopted open source solutions.

Regarding the management plane (MP) of SDN networks, there are several proposed reference models such as TAPI (Transport API) [28], ACTN (Abstraction and Control of Traffic engineered Networks) [29], and COP (Control Orchestration Protocol) [30] that abstract the control plane functions of the SDN network. These reference models typically define a basic set of management services (e.g., topology service, connectivity service, inventory service, path computation service, virtual network service, and notification service) to facilitate the automation of the SDN network operation, deal with the heterogeneity of NBIs, and realize the concept of Transport Network as a Service (TNaaS). By way of illustration, the ACTN framework is a set of management a control functions defined by the IETF Traffic Engineering and Signalling Working Group. ACTN enables the abstraction of the underlying physical topology of the SDN forwarding plane and cross-domain coordination within the SDN networks. In this way, ACTN enables the customer to create and operate virtual transport networks, while hiding the complexity of the underlying infrastructure. There is an ongoing project for developing ACTN reference model with ONOS [31].

Last, it is noteworthy to mention, there are several private network scenarios such as Industry 4.0 demanding deterministic Quality of Service (QoS). In this vein, IEEE 802.1Q Time-Sensitive Networking (TSN) standards appear as a promising option. The TSN standard is a converged Layer 2 network technology able to provide stringent deterministic QoS in terms of reliability, E2E latency and jitter, and frame loss. In the same way, Deterministic Networking (DetNet) is an IETF ongoing standardization effort to ensure deterministic QoS in Layer 3. TSN and DetNet define SDN like architectures for controlling the DP network devices. There are proposals in the literature for the integration of the TSN centralized CP and standard SDN.

2.2 Orchestration frameworks for private networks

2.2.1 State of the art

NFV MANO (network functions virtualization management and orchestration) is an architectural framework for managing and orchestrating virtualized network functions (VNFs) and other software components [32]. The European Telecommunications Standards Institute (ETSI) Industry Specification Group (ISG NFV) defined the MANO architecture to facilitate the deployment and connection of services as they are decoupled from dedicated physical devices and moved to virtual machines (VMs). The focus of NFV MANO is highlighted in Figure 2.3. NFV MANO is broken up into three main functional blocks:

- **NFV Orchestrator (NFVO)** is responsible for: 1) on-boarding of new network service, VNF forwarding graph and VNF packages, 2) NS lifecycle management (including instantiation, scale-out/in, performance measurements, event correlation, termination), 3) global resource management, validation and authorization of network functions virtualization infrastructure (NFVI) resource requests and 4) policy management for NS instances.
- **VNF Manager (VNFM)** is responsible for lifecycle management of VNF instances, and overall coordination and adaptation role for configuration and event reporting between NFVI and the traditional element / network management system (E/NMS).
- **Virtualized Infrastructure Manager (VIM)** is responsible for controlling and managing the NFVI compute, storage and network resources, within one operator's infrastructure sub-domain, as well as collection and forwarding of performance measurements and events.

These blocks are responsible for deploying and connecting functions and services when they are needed throughout the network. Such capabilities allow to launch and manage E2E services in minutes, instead of hours. It also enables a new range of features such as agile service provisioning, multitenancy, software controlled and dynamic management, on demand service-oriented resource allocation, universal multi access, and integration and interoperability among different network resources. In this ecosystem, the NFV MANO framework is an essential enabler to ease and realize 5G vision.

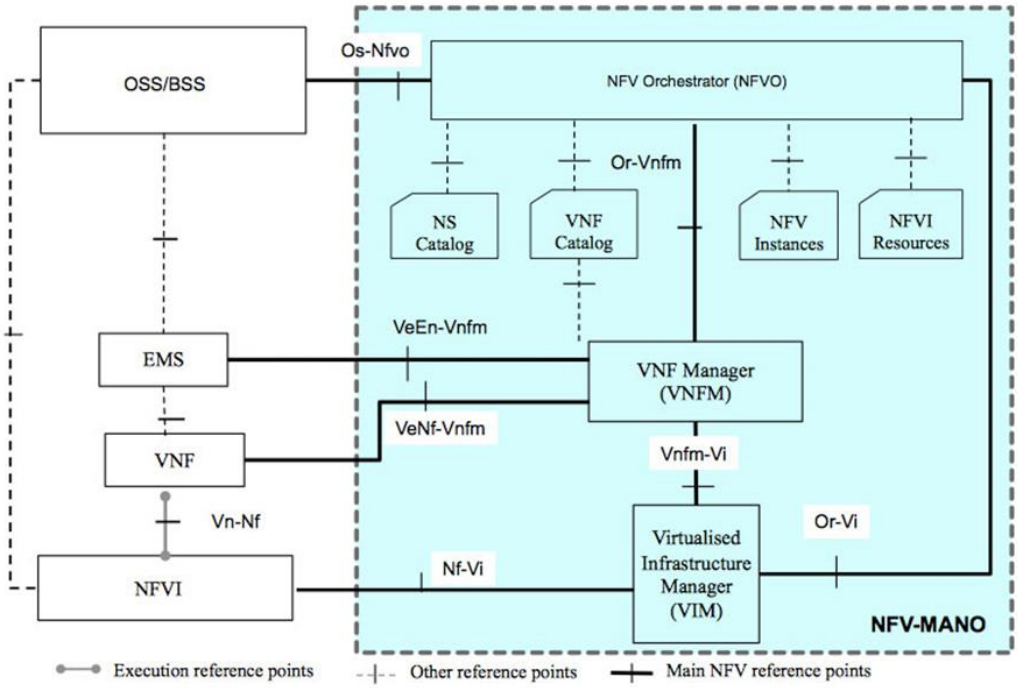


Figure 2.3 NFV MANO framework.

To handle its responsibilities the NFV MANO framework includes a group of repositories, namely NS catalog, VNF catalog, NFV Instances and NFVI resources. In addition to MANO internal functional blocks, there are two important external elements, the EMS and OSS/BSS, continuously exchange information with the MANO framework in order to meet the expected operational requirements.

NFV entities to deploy and manage by MANO framework are listed below:

- **Network Service (NS**) described by its descriptor file, orchestrated by NFVO. It may cover 1 or more VNF Graphs, VNFs and Physical Network Functions (PNFs)
- **VNF Forwarding Graph (VNF-FG)** described by its descriptor file, orchestrated by NFVO. It may cover VNF-FGs, VNFs and NFs.
- **Virtualized Network Function (VNF)** described by its descriptor file, instantiated by the VNF Manager. It covers VNF components (VNFC) each mapped to a VM described with the Virtual Deployment Unit descriptor.

Information in the NFV entities is structured into information elements, which may contain a single value or additional information elements that form a tree structure. Information elements can be used in two different contexts: as descriptors or as run-time instance records. Next, we describe the main descriptors defined by ETSI NFV:

- **Network Service Descriptor (NSD)** is referencing all other descriptors, which describe components that are part of that NS.
- **VNF Descriptor (VNFD)** describes a VNF in terms of its deployment and operational behaviour requirements. It also contains connectivity, interface and virtualized resource requirement.
- **VNF Forwarding Graph Descriptor (VNFFGD)** describes a topology of the NS or a portion of the NS, by referencing VNFs and PNFs and Virtual Links that connect them.
- **Virtual Link Descriptor (VLD)** describes the resource requirements that are needed for a link between VNFs, PNFs and endpoints of the NS, which could be met by various link options that are available in the NFVI.

- **Physical Network Function Descriptor (PNFD)** describes the connectivity, interface and KPIs requirements of virtual links to an attached physical network function.
- **Virtualisation Deployment Unit (VDU)** describes the deployment and operational behaviour of a subset of a VNF, or the entire VNF if it was not segmented in subsets. A VDU is deployed as a VM in the VNF.

ETSI NFV MANO Orchestration Platform

Next, existing ETSI NFV MANO orchestration platforms are analysed to see if there is possibility to adopt them in the specific context of 5G-CLARITY. The main ETSI NFV MANO orchestration platforms are:

Open Baton is an open source platform that provides a comprehensive implementation of the ETSI NFV MANO specification [33]. The main features and components of Open Baton are 1) a NFVO; 2) a generic VNFM that manages VNF life cycles based on the VNF description; 3) an Auto-scaling engine which can be used for automatic runtime management of the VNFs; 4) a fault management system for automatic management of faults; 5) an SDK comprising a set of libraries that could be used for building a specific VNFM; and 6) a dashboard for managing the VNFs. Open Baton is currently using OpenStack as first integrated NFV point of presence (PoP) VIM, supporting dynamic registration of NFV PoPs and deploys in parallel multiple slices, one for each tenant, consisting of one or multiple VNFs. Through this functionality, the orchestrator provides a multi-tenant environment distributed on top of multiple cloud instances.

Cloudify is an open source TOSCA-based orchestration platform [34], which is designed to fit as an NFVO and a generic VNFM. VNFs and PNFs are modelled using the TOSCA language and on-boarded to Cloudify. Cloudify model driven Design allows operators to build VNF descriptors and network service descriptors and manage the lifecycle of the network service. By its very nature, Cloudify Blueprints might be considered as the NS and VNF catalog entities defined by MANO. Cloudify is aligned with the MANO reference architecture but not fully compliant.

Tacker is an official OpenStack project [35] that builds a generic VNFM and a NFVO to deploy and operate network services and VNFs on an NFV infrastructure platform like OpenStack (at this moment, the multi-VIM is not supported, as Tacker only supports OpenStack). It is based on ETSI MANO architectural framework and provides a functional stack to Orchestrate Network Services E2E using VNFs.

Open Network Automation Platform (ONAP) [36] provides a platform for real-time, policy-driven orchestration and automation of physical and virtual network functions, which allows providers and developers to automate new services and support lifecycle management. The orchestration stack on ONAP provides for service delivery, change, scaling controller instantiation and capacity management across both the application and network controllers.

Open Source MANO (OSM) [37] is an operator-led ETSI community effort that is delivering a production quality open source Management and Orchestration (MANO) stack aligned to the ETSI NFV information models and that meets the requirements of production NFV networks. OSM provides a solution for onboarding and instantiation of network services including VNFs for different use cases in the virtual and physical network elements. It is also covering all the aspects of network service life-cycle management and network function life-cycle management. OSM takes a huge step towards 5G network deployments and their E2E orchestration by telecom operators, making it an ideal choice for 5G scenarios and any kind of Network as a Service (NaaS) offer. The main and extended features of OSM are:

- Provides a python-based client library and REST interfaces to enables access to all features, as well as a set of plugin models for integrating multiple monitoring tools and SDN controllers.
- Provide dynamic creation of inter-datacentre connections across heterogeneous Wide-Area Networks (WANs).
- Bringing complete support of 5G network slices for the cloud domain and cloud-native applications

to NFV deployments.

- Expanding orchestration capabilities across transport, physical and hybrid networks.
- Extended support of Service Function Chaining (SFC) monitoring capabilities, including VNF metrics collection; and support for physical and hybrid network functions, (PNFs and HNFs respectively).

Summaries of the solutions demonstrate that most of existing orchestration tools are limited to the cloud domain and not dealing with RAN and transport network. In addition, the slicing and cloud-native applications are not tackled, so they do not explicitly handle solutions in the context of 5G-CLARITY project. Among others, the project has chosen to utilize the ETSI OSM for the orchestration as it is the only solution that provides required features to our project.

Edge Computing Orchestration Framework

Edge computing is an extension of cloud architectures to the edge of the network, close to the devices that produce and act on data. The value proposition of this new model is that: *i)* it can process data close to where they are collected thus minimizing processing latency; *ii)* it offloads gigabytes of network traffic from the core network; and, *iii)* it keeps sensitive data inside the network itself. Like in the cloud domain, edge resources require to be orchestrated and managed. Here, we are reviewing some orchestration frameworks that have been designed for edge computing:

Open Network Edge Services Software (OpenNESS) is an open source software toolkit that enables the deployment of edge compute services on diverse platform and access technologies [38]. It supports the deployment of edge services in a network and can be extended in functionality into a commercial platform or be re-used to add capability to existing edge platforms. OpenNESS is designed to reduce the "deployment impedance" experienced by network operators, independent hardware vendors (IHVs), and independent software vendors (ISVs) in deploying edge services. To achieve this goal, OpenNESS provides a variety of features such as:

- Exposes platform and hardware diversity to edge applications and orchestrators.
- Supports the extension of public cloud services into the edge network, as well as the RAN technologies in the edge networks, including wire, Wi-Fi, LTE and 5G mobile networks.
- Supports Artificial Intelligence (AI) and media computing application frameworks.
- Supports edge applications and network edge application onboarding, as well as core networks.
- Supports VM deployments on Kubernetes, OpenStack.

ONF Aether is an open source Enterprise 5G/LTE Edge-Cloud-as-a-Service platform (ECaaS) [39], which provides mobile connectivity and edge cloud services for distributed enterprise networks, all provisioned and managed from a centralized cloud. Aether is optimized for multi-cloud deployments, and it simultaneously supports wireless devices over licensed, unlicensed and lightly licensed (CBRS) spectrum. It also can support mobility and edge services across a multi-site footprint as an elastic and scalable cloud service, simplifying deployment. Management is centralized in the cloud providing enterprise-wide visibility and a centralized dashboard for management.

The described edge computing orchestration frameworks do not adhere to the NFV MANO architecture and information elements and therefore will not be considered in 5G-CLARITY , given that our focus is designing management frameworks for private networks that can be easily integrated with 5G public networks where NFV MANO is being used.

2.3 Network slicing in private networks

2.3.1 State of the art

2.3.1.1 Legacy pre-slicing techniques

Network slicing is a new concept brought in holistically within the new 5G paradigm in the Third Generation Partnership Project (3GPP) specification from Release 15 (Rel-15) onwards. However, to a very reduced and lesser extent, there have been certain network sharing and pre-slicing aspects since the early days of 3GPP Rel-9 specification, albeit more related to Quality of Service (QoS) management or network sharing aspects.

a) QoS Class Identifiers (QCIs) and Guaranteed Bit Rate (GBR)

In [40] the LTE Rel-9 specifications already specified the concept of E2E QoS by means of the Evolved Packet System (EPS) bearer/ EPS-Radio Access Bearer (E-RAB) for E2E QoS control in the EPC/E-UTRAN architecture. The Service Data Flows (SDFs) mapped to the same EPS bearer receive the same bearer level packet forwarding treatment (e.g. scheduling policy, queue management policy, rate shaping policy, Radio Link Control (RLC) configuration, etc.). An EPS bearer/E-RAB is referred to as a GBR bearer if dedicated network resources related to a GBR value that is associated with the EPS bearer/E-RAB are permanently allocated at bearer establishment/modification. Otherwise, an EPS bearer/E-RAB is referred to as a Non-GBR bearer. This EPS bearer service layered architecture is depicted in Figure 2.4.

The bearer level QoS parameters are QCI, Allocation and Retention Priority (ARP), GBR, and Aggregate Maximum Bit Rate (AMBR). The QCI is a scalar that is used as a reference to access node-specific parameters that control bearer level packet forwarding treatment. There is a one-to-one mapping of standardized QCI values to standardized characteristics. The ARP primary purpose is to decide whether a bearer establishment / modification request can be accepted or needs to be rejected in case of resource limitations. Each GBR bearer is additionally associated with the GBR, which is the bit rate that can be expected to be provided by a GBR bearer. E2E resource allocations for GBR bearers are considered for admission control of new bearers to guarantee that bitrate. To an extent, the use of GBR and ARP could be considered as a primitive pre-slice resource reservation technique. While GBR is used to guarantee the minimum data rate to be maintained to support by a single bearer, the User Equipment (UE)-AMBR and Access Point Name (APN)-AMBR are the maximum aggregated bitrates associated with a group of bearers per UE or APN respectively.

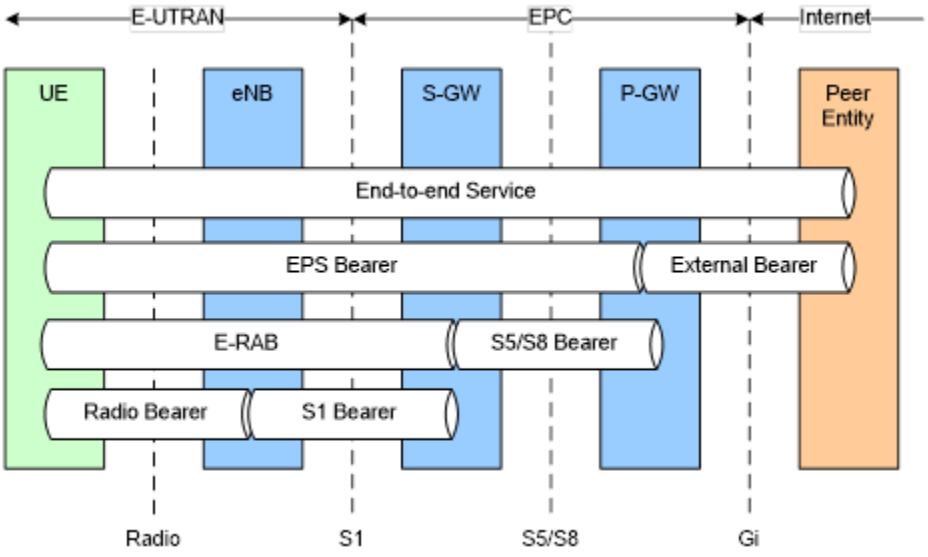


Figure 2.4: EPS bearer service architecture.

b) APN

From [41] a UE can be connected to more than one Packet Data Network (PDN) for different types of services, for example typically UEs can use different APN associated to the different PDNs such as "internet" for normal internet or "ims" for VoLTE services. This possibility to separate services on an APN basis could also be seen as a way to "slice" the services of the network. Within the APN, the above E2E QoS framework is applied.

c) Network Sharing: Multi-Operator Core Network (MOCN)

In [42] the defined network sharing architecture allows different core MNOs to connect to a shared radio access network. This network sharing can either be implemented as Multi-Operator Core Network (MOCN), i.e., with Non-Access Stratum Node Selection Function (NNSF) within the eNodeB, or GateWay -GateWay Core Network (GWCN), i.e. with NNSF within the EPC. MOCN enables not only the possibility to support neutral host use cases where the eNodeB connects different MNO subscribers to each participating MNO EPC based on the PLMNID used during registration, but also the support of pure S1-Flex configurations for load sharing between different pools of Mobility Management Entities (MMEs) and Serving Gateways (SGWs) for single MNO use cases. Multi-operator RAN (MO-RAN) is a special case of network sharing whereby the Radio Units (cells) are fully dedicated to each MNO normally using their own licensed spectrum, but all the underlying mechanisms used to support it (e.g. broadcasting multiple PLMNIDs, NNSF, etc.) are the same as for MOCN.

d) Dedicated core network selection function (DECOR)/enhanced DECOR (eDECOR)

DECOR was part of 3GPP Rel-13 [43] and 3GPP Rel-14 (eDECOR) to provide a finer grained granularity for differentiated services within a PLMNID [44]. DECOR enabled the EPC selection function within the PLMNID for different types of subscribers associated to different Dedicated Core Networks (DCN). The purpose with a dedicated core network may be to provide specific characteristics and/or functions or isolate specific UEs or subscribers, e.g. machine-to-machine (M2M) subscribers, subscribers belonging to a specific enterprise or separate administrative domains, etc. The main architecture enhancements are to route and maintain UEs in their respective dedicated core network, i.e. for UEs with assigned DCN. DECOR introduced UE usage type parameter in Home Subscriber Server (HSS) and signalling procedures to define which DCN (i.e. service) a specific UE should be attached to, within the same PLMNID. eDECOR enhanced DECOR with assistance data to UE so that Non-Access Stratum (NAS) Initial UE message re-routing is not needed.

2.3.1.2 Network slicing concept

Network slicing has emerged as a solution to economically provide the tenants' use cases over a common infrastructure. For this reason, the main Standard Developing Organizations (SDO) have taken the leadership role on analysing the concept of network slicing as well as standardizing its management and orchestration mechanisms. A brief description of SDO contributions on network slicing can be found in 5G-CLARITY deliverable D2.2 [45]. Below, we provide an insight into the network slicing contributions related to multi-Wireless Access Technology (WAT) networks and private networks. Finally, we also provide a brief description about the latest advances on slicing.

2.3.1.3 Radio Access Network (RAN) slicing for multi-WAT networks

Thanks to network slicing, 5G systems are expected to be flexible infrastructures where slices are created with appropriate isolation and optimized characteristics to address the requirements of a specific application. This is especially relevant for the RAN, where resources are expensive, scarce and cannot be overprovisioned. Focusing on RAN slicing, the slices run on top of the wireless platform which contains radio hardware and the set of infrastructure resources. A RAN slice can be seen as a particular RAN behaviour, and it has to be designed in order to meet specific RAN capabilities and networks characteristics required by different

services associated with a given Single-Network Slice Selection Assistance Information (S-NSSAI) and Public Land Mobile Network (PLMN). According to [46], network slicing in RAN must concern about a series of key principles such as RAN awareness of slices to make distinctions in traffic handling in accordance to different network slices requirements; the slice availability; and the correct resource management and isolation between them. Different works available in the literature have addressed the isolation and the radio resource management between RAN slices [47], [48].

From a functional perspective of RAN slices, their radio functionalities are gathered in gNodeB (gNBs). These gNBs split their functionalities into Distributed Units (DUs) and Centralized Units (CUs). The DU comprises lower-layer functionalities and might be partially implemented as VNFs in micro servers while the CU comprises higher-layer functionalities and might be totally implemented as VNF in an edge cloud. This functional split allows to benefit from the processing centralization and ease the aggregation of multi-WAT, e.g. Wi-Fi and LiFi, over common functionalities.

Among the existing works in addressing RAN slicing in non-3GPP access networks, [52] introduces a resource slicing scheme which realizes autonomic management and configuration of virtual Access Points (AP) in a LiFi attocell access network based on service providers and their user service requirements. This scheme comprises of traffic analysis and classification, a local AP controller, downlink and uplink slice resource manager, traffic measurement, and information collection modules. The proposed resource slicing scheme collects and analyses different applications traffic statistics supported on the slices defined in each LiFi AP and distributes the available resources fairly and proportionally among them. In [53], RAN slicing in Wi-Fi networks is addressed where a scheduling algorithm that allocates airtime to a set of virtual interfaces executing on the same or in different physical radios is developed.

5G-CLARITY will advance the SotA by considering network slices supporting radio access technologies beyond 5GNR, i.e., Wi-Fi and LiFi. Furthermore, 5G-CLARITY goes beyond the SotA by developing a RAN slice orchestrator able to cope with radio frequency resources (licensed and unlicensed spectrum) and LiFi resources. 5G-CLARITY slices will have to deal with multi-connectivity features supporting the Access Traffic Steering, Switching and Splitting (AT3S) function, which oversees traffic steering between 3GPPP and non-3GPP accesses [54].

2.3.1.4 Network slicing solutions for private networks from 5G-PPP projects

Working in parallel with SDOs, 5G-PPP Phase 3 projects consider the SDO solutions for network slicing and provide novel contributions to that field. Within this phase, two groups are relevant to the 5G-CLARITY scope as shown in Figure 2.5, *a)* infrastructure projects (ICT-17 call), i.e. 5G-VINNI, 5G-EVE and 5GENESIS, and *b)* vertical use case projects (ICT-19 call), i.e. 5GSMART, 5G-TOURS, FULL5G, 5G!DRONES, 5GROWTH, 5G-HEART, 5GSOLUTIONS, 5G-VICTORI [1]. On the one hand, ICT-17 projects provide a pan-European large-scale 5G validation network infrastructure that demonstrates how key 5G Key Performance Indicators (KPIs) can be met, accessed and used by industry verticals to set up research trials of innovative use cases, testing and validating specific applications that depend upon those KPIs. On the other hand, ICT-19 projects aim the technical and business validation of 5G technologies from the verticals' viewpoint, following a field-trial-based approach on vertical specific private venues, e.g., factories, campus, etc.

Figure 2.5: Overview of relevant 5G-PPP Phase III projects, and their relation as "public" (ICT-17) and "private" (ICT-19) 5G networks [1].

Under this context, the network defined within the logical perimeter of a vertical-owned site (i.e., private network) usually consists of one (or more) network domain(s). The rest of the domains thus shall be provided an external network (i.e., public network) out of the logical perimeter of the vertical premises. To integrate both, the public and non-public networks, network slicing takes a key role. Specifically, with an ICT-17 site providing a given ICT-19 site with missing 5G components (e.g., 5G control plane, ICT-17 support application) and necessary connectivity (e.g., WAN connectivity) in the form of a dedicated Network Slice Subnet Instance (NSSI). By attaching this NSSI to the on-premises deployed components, the ICT-19 site can have an E2E Network Slice Instance (NSI) at his disposal. An illustrative example of this integration is depicted in Figure 2.6.

Focusing on the network slicing solutions provided by ICT-19 projects, most of them are in their infancy because only abstracted concepts have been developed at the moment of writing this deliverable. Other ICT-19 projects such as 5G-TOURS and 5GROWTH have already provided more exhaustive details about their solutions for network slicing. Specifically, 5G-TOURS provides an efficient and reliable close-to commercial services for tourists, citizens and patients in three different types of cities: (a) the safe city where e-health use cases will be demonstrated; (b) the touristic city focused on media and broadcast use cases; and (c) the mobility-efficient city that brings 5G to users in motion as well as to transport related service providers [55]. To provide the vertical customers with a friendly interface to request and operate network slices, 5G-TOURS implements a service layer on the top of the management architecture. Specifically, this layer includes management mechanisms such as slice definition, creation, monitoring management and deletion by means of network slice blueprints. 5G-CLARITY will use the 5GT Vertical Slicer component from the 5G-Transfomer project (i.e., a 5G-PPP Phase 2 project) as a baseline to implement the network slice blueprint and the service layer [56].

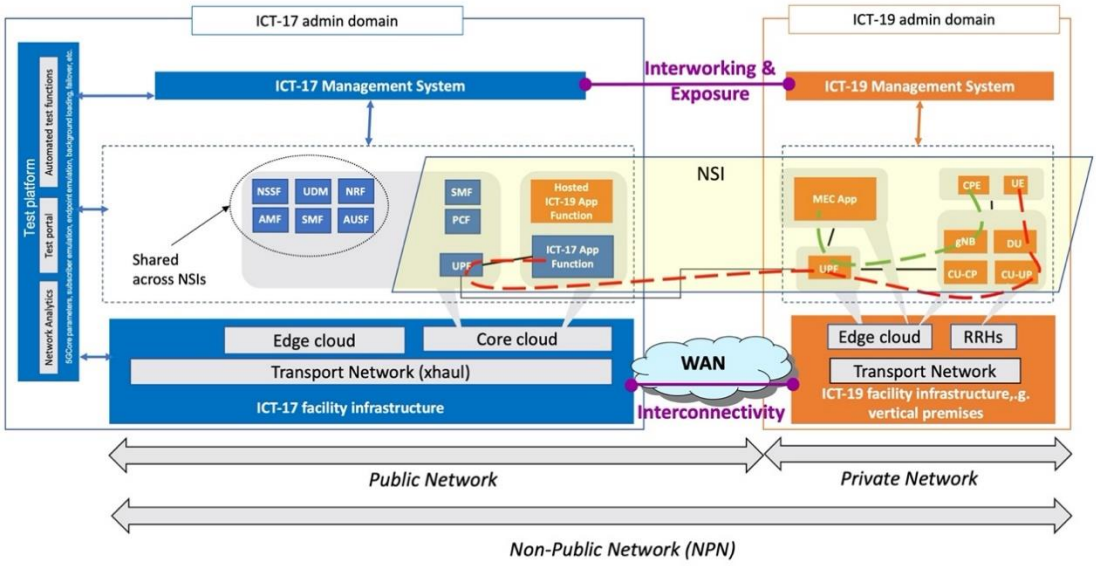


Figure 2.6: Example of public and private network integration with network slicing (5G-CLARITY, 2020).

5GROWTH aims to enable the uniform and automated deployment and operation of customized network slices for emerging use cases (e.g., Industry 4.0, transportation and energy on vertical premises). As the starting point, 5GROWTH inherits the 5G-TRANSFORMER architecture for managing network slices [57]. 5GROWTH architecture is composed of three layers: 5GROWTH vertical slicer (5Gr-VS), 5GROWTH service orchestrator (5Gr-SO), and 5GROWTH resource layer (5Gr-RL). 5Gr-VS acts as entry point for verticals to request a custom network slice. It manages the mapping and translation between the requested vertical services (selected from the catalogue of Vertical Service Blueprints (VSB)) and a number of NSIs that are created on demand, by provisioning the associated NFV network services mediated through the 5Gr-SO. 5Gr-VS also implements the functionalities to manage the lifecycle of the network slices and their network slice subnets that can be mapped into the Network Slice Management Function (NSMF) and Network Slice Subnet Management Function (NSSMF). 5Gr-SO is responsible for E2E orchestration of NS and lifecycle management. It provides both NS and resource orchestration capabilities to instantiate network slices within and across multiple domains. 5Gr-RL hosts all the compute, storage and networking physical and virtual resources where network slices and E2E services are executed.

In a nutshell, the ICT-19 projects assume an integration of public and private networks to provide use cases required by the industry verticals. In these projects, network slicing is the key enabler to perform this integration. Furthermore, in order to automate the deployment and operation of the necessary network slices, AI and Machine Learning (ML) play a key role. In 5G-CLARITY, we do not only adopt these innovations, but we also give a step further by considering a scenario where the private venues have available multi-WAT, i.e., 5GNR, Wi-Fi and LiFi. The aim of using multi-WAT is to enhance the available capacity in the vertical premises [58].

2.3.1.5 Latest advances on network slicing

Some of the latest advances in network slicing such as multi-domain slicing, hybrid slicing and deep slicing are introduced below.

a) Multi-domain slicing

Multi-domain slicing refers to the slicing techniques that span across all the network domains, including RAN, core and transport domain. The network slices that embrace all the network domains are typically referred

to as E2E slices.

In recent years, several solutions for network slicing were introduced in the literature, however, most of them focus on single domains. Specifically, early solutions have been focused on the core network, (e.g., [59]). Then came slicing solutions for the RAN and transport networks (e.g., [60] for RAN slicing and [61] for transport network slicing). However, neither of these solutions contemplate a network slice as an E2E solution. For this reason, the last advances on network slicing by the research community are focused on E2E solutions on several aspects. For example, in [62] POSENS is designed, which is a practical open source solution for E2E network slicing. This solution extends the SotA proposals by including in the prototype characteristics such as multi-slice UE, slice-aware RAN solution, and specific multi-slice MANO capabilities. Other works further focus on improving technical aspects on the management architecture for E2E slicing. For instance, [63] provides an implementation of an E2E network slice orchestration platform aiming at the evaluation of its performance in terms of dynamic deployment of network slices in an E2E fashion. They also discuss how the slice orchestrator functionality can be enhanced to better customize the network slices according to the needs of their respective service. Finally, some works provide mathematical frameworks to manage and orchestrate E2E slices, such as [64], in which a model to build a network slice request and map it into the infrastructure network is presented. The mapping process consists of the placement of VNFs and selection of link paths chaining them. In this solution, a complex network theory to obtain the topological information of slices and infrastructure network is also adopted.

b) Hybrid slicing and deep slicing

Hybrid slicing is a concept used to refer to those scenarios where the available radio resources are not allocated in static portions to the different slices. This means there is an amount of resources that is shared between slices that offer different kind of services as demonstrated in Figure 2.7. The distribution of these shared resources must be optimal according to the needs of the existing slices throughout their lifetime. Under this context, a dynamic resource allocation framework to facilitate RAN slicing among heterogeneous services is proposed [65]. It aims at jointly optimizing the bandwidth allocation and power consumption for IoT devices while satisfying the corresponding latency for sporadic ultra-Reliable Low Latency Communication (uRLLC) traffic arrivals and the quality of enhanced Mobile Broadband (eMBB) services as much as possible. Some radio resources are reserved for uRLLC and eMBB slices in advance, while others are shared according to the queue backlog to customize the frequency bandwidth slice.

The concept of deep slicing was introduced in [66]. It refers to have any kind of service, any resource and any function wherever, in such a way that multiple stakeholders can be supported using one infrastructure. To achieve this goal, all the novel slicing techniques and technologies must be involved, such as E2E slicing, multi-domain slicing and proper slices scaling and isolation.

5G-CLARITY will consider E2E slices in two different ways. First, the 5G-CLARITY management stratum described in deliverable D2.2 [45] will be able to deploy multi-domain slices within the private venue composed of multi-WAT, transport and compute services. Second, the 5G-CLARITY slices within the private venue may be complemented with slices offered over a public network.

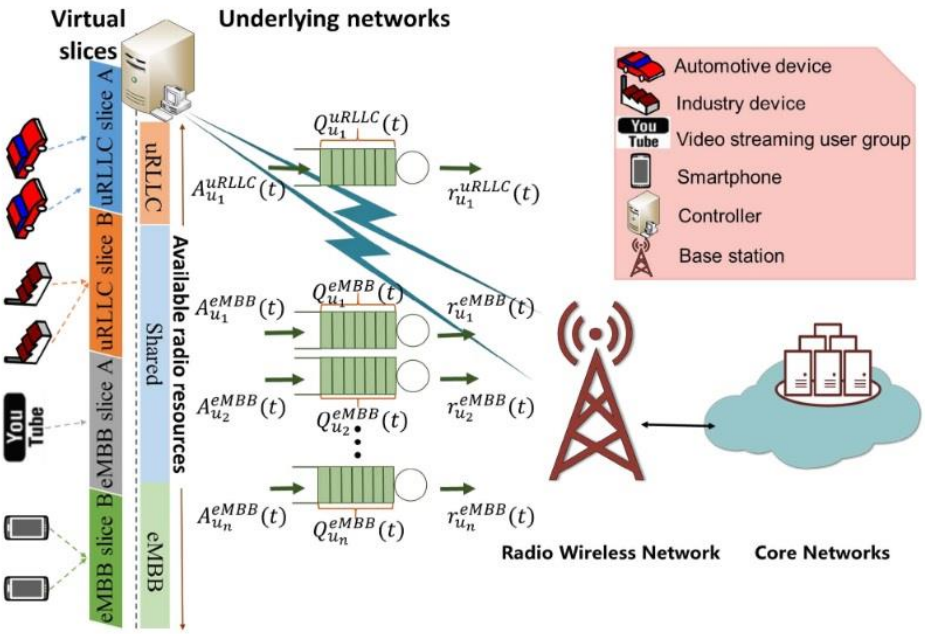


Figure 2.7: Radio resource allocation for hybrid services. Figure taken from [65].

2.3.2 5G-CLARITY slicing model and realization

A **5G-CLARITY slice** is defined as an on-premise infrastructure slice, i.e., an isolated set of resources that are segregated from the private infrastructure for their delivery to separate tenants. This means 5G-CLARITY slicing enables isolation among tenants. Note that the use of network slicing in 5G-CLARITY differs from the one in 3GPP community, i.e., while for 5G-CLARITY it is **multi-tenancy support**, for 3GPP it is **multi-service support** (the ability to deliver tailored functionality for different services). Once a single 5G-CLARITY slice is delivered to a specific tenant, one or more customer-facing services could be provisioned using the set of resources (hereinafter referred as resource chunks) which comprises this 5G-CLARITY slice. These customer-facing services (augmented reality applications, enhance positioning automated guided vehicles) could be modelled as one or more ETSI-NFV network services, i.e., a composition of NFs implemented as VNFs and a set of wireless and transport services configured on PNFs, as they will be defined later. The delivered 5G-CLARITY slice could be deployed across different network domains, i.e., RAN Transport Network (TN) or Core Network (CN). This means the 5G-CLARITY slice has an E2E nature.

Depending on the intended use of individual 5G-CLARITY slices, a tenant could be either:

- A Communication Service Provider (CSP) / Digital Service Provider (DSP), either public or private: It uses the 5G-CLARITY slice as received from the private network operator, without further modifications. The CSP/DSP simply builds communication services on top of the slice. With this setup, the services are entirely provisioned within the boundaries of the private venue.
- A public network operator: It uses the 5G-CLARITY slice received from the private operator to define a new slice, with further functionality. The public network operator can then re-sell this slice to public or private CSP/DSPs, fostering B2B2X partnerships. The process of defining a new slice out of the 5G-CLARITY slice can be done using two different approaches. On the one hand, the public network operator can deploy public VNFs on the private infrastructure, using the in-house resources managed by the private network operator, and attach these VNFs to the original 5G-CLARITY slice. Note that the resulting slice is also on the on-premise slice. On the other hand, the public operator can extend the original 5G-CLARITY slice to the PLMN, where the public operator can do further processing (e.g., aggregating public VNFs, or extend coverage). Unlike the first scenario, the resulting

slice is not an on-premise slice, as it includes PLMN resources. These off-premise resources are beyond the management scope of private network operator, and therefore of 5G-CLARITY system.

- An hyper-scaler: It performs an equivalent tenant role as the public network operator, with the only difference that off-premise resources are not PLMN resources, but private cloud resources, e.g., Google Cloud, Amazon Web Services, Microsoft Azure. The consideration of this tenant is out of 5G-CLARITY scope.

Focusing on the resource chunk composition of a 5G-CLARITY slice, it can be seen as a tailored set of 5G-CLARITY resource-facing services, i.e., *a)* one or more 5G-CLARITY wireless services; *b)* one or more 5G-CLARITY compute services; and *c)* one or more 5G-CLARITY transport services. These individual services could be flexibly combined to define a wide range of different 5G-CLARITY slices. These services are individually detailed below.

A **5G-CLARITY wireless service** represents the configuration that needs to be set on one or more APs or base stations from a specific WAT, in order to make them operationally ready for one 5G-CLARITY slices. Additionally, different WATs require the definition of separate 5G-CLARITY wireless services, thus the number of 5G-CLARITY wireless services in a 5G-CLARITY slice ranges from one to a maximum of three, i.e., LTE/5GNR, Wi-Fi and/or LiFi. Figure 2.8 shows an example of 5G-CLARITY slice with three different 5G-CLARITY wireless services. If the WAT of a 5G-CLARITY wireless service consists of LTE or 5GNR APs, i.e., eNBs or gNB-DUs, these must be configured to radiate a given PLMN Identifier (PLMN ID). When the WAT of a 5G-CLARITY wireless service consists of Wi-Fi or LiFi APs, these APs must be configured to radiate a given Service Set Identifier (SSID).

A **5G-CLARITY compute service** is a composition of VNFs, each providing a well-defined network functionality. This means a 5G-CLARITY compute service can be modelled as a single ETSI-NFV NS. Among the VNFs which a 5G-CLARITY slice comprises, a subset of them must implement the 3GPP functionality that is required to provide connectivity from end-user devices to a data network. The number of 5G-CLARITY compute services required by a given 5G-CLARITY slice depends on the modularity in the 5G-CLARITY slice design. The higher the modularity, the higher the number of compute services needed. For example, the VNFs providing the control plane functionalities of 5G CN, the UPF (along with the correspond vAPP) and the gNB-CU (along with the dRAX) could be modelled as separate 5G-CLARITY compute services (NSs provisioned each on a single site), or could be arranged into one single 5G-CLARITY compute service (a NS provisioned over multiple sites), or any combination in between. The former case is depicted in Figure 2.8. In this example, one network service deployed in the RAN cluster implements the gNB-CUs and the dRAX for a 5G-CLARITY slice. Other three network services deployed in the Edge cluster implement the CN control plane, and two UPFs (along with their vAPPs), respectively.

A **5G-CLARITY transport service** represents the configuration that needs to be set on TN devices in order to make them deliver the traffic from one or more 5G-CLARITY wireless services into one or more 5G-CLARITY compute services. TN could comprise Ethernet and/or Time-Sensitive Networking (TSN) switching devices. In 5G-CLARITY, the frames of each transport service will be uniquely signalled by using one or more 802.1Q VLAN tags.

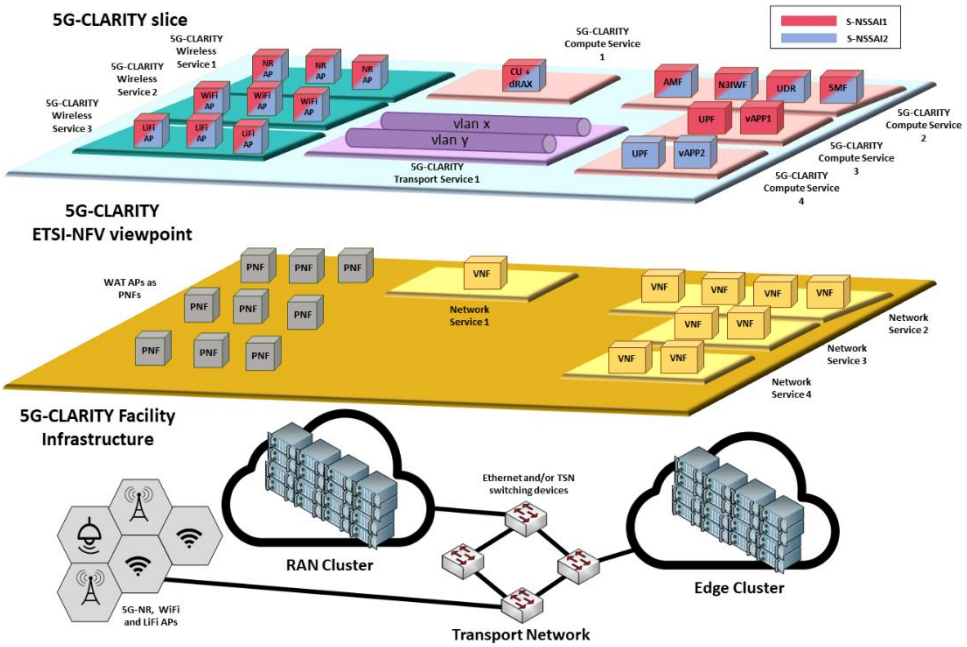


Figure 2.8 Deployment example of an on-premise 5G-CLARITY slice.

For a given 5G-CLARITY wireless/compute/transport service within a 5G-CLARITY slice, it is required to define the specific size for the resource chunk to be allocated for that 5G-CLARITY service. This amount of resources is referred to as resource quota**.** The definition of resource quotas for each 5G-CLARITY slice ensures their proper isolation during their lifetime. The definition of resource quota for each 5G-CLARITY slice must be detailed for each constituent 5G-CLARITY wireless/compute/transport service. We describe the meaning of resource quota for each 5G-CLARITY service as follows.

A **5G-CLARITY wireless quota** is the set of wireless resources (i.e., averaged over a time interval) which are allocated in each AP of a specific 5G-CLARITY wireless service. This quota could be strict or float. A strict quota means wireless resources are not allowed for other 5G-CLARITY slices even when they are not used by the defined 5G-CLARITY slice. A float quota means wireless resources could be allowed for other 5G-CLARITY slices if they are not used by the defined 5G-CLARITY slices. When the WAT supported by the wireless service consists of LTE/NR APs, the wireless resources correspond to Physical Resource Blocks (PRBs). If the WAT supported by the wireless service consists of Wi-Fi APs, the wireless resources correspond to the percentage of airtime consumption. Finally, if the WAT supported by the wireless service consists of LiFi APs, the wireless resources are specific wavelengths and/or airtime on a wavelength. The 5G-CLARITY wireless quotas will be enforced through the multi-WAT non real time Controller (see section 2.3.2.1.1).

A **5G-CLARITY compute quota** is the set of virtualized computing, networking and/or storage resources from cloud infrastructures (e.g., the RAN and/or edge cluster) which are allocated for a 5G-CLARITY slice.

- For virtualized computing resources, the 5G-CLARITY compute quota comprises a restriction on the number of virtualized CPUs, the number of virtualization containers (e.g., virtual machines) or the size of the virtual memory (i.e., virtualized RAM).
- For virtualized networking resources, the 5G-CLARITY compute quota comprises a restriction on the number of public IP addressed, the number of ports and the number of subnets.
- For virtualized storage resources, the 5G-CLARITY compute quota includes a limitation on the storage size, the number of snapshots and the number of volumes.

To implement the 5G-CLARITY compute quota for each 5G-CLARITY compute service, a VIM project (i.e., OpenStack project) will be created.

A **5G-CLARITY transport quota** is the set of transport connectivity resources that can be allocated to a 5G-CLARITY transport service. The resources available for use depend on the underlying transport technology, and could be expressed in different terms, including data-rate, latency or buffer space. To signal a 5G-CLARITY transport quota for each 5G-CLARITY transport service, tools such as Control Orchestration Protocol (COP) or Transport API (TAPI) will be used.

Finally, the definition of 5G-CLARITY slice comprises a set of service identifiers as:

- One or more S-NSSAIs. Each S-NSSAI is used to connect the UEs with a specific customer-facing service. Figure 2.8 shows an example of a 5G-CLARITY slice using two S-NSSAIs, i.e., 5G-CLARITY slice 1 using S-NSSAI1 and S-NSSAI2, each used for connecting UEs with the vAPP1 and vAPP2, respectively.
- One PLMN ID. This mandatory identifier identifies the operator which provide the 5G-CLARITY slice, i.e., the private owner or the public operator.
- One or more Service Set Identifiers (SSIDs). Each SSID identifies a specific customer-facing service for those UEs connected to the 5G-CLARITY slice by a non-3GPP 5G-CLARITY wireless service (i.e., Wi-Fi and LiFi). This means a 5G-CLARITY slice requires one SSID per S-NSSAI if a non-3GPP 5G-CLARITY wireless service offers this customer-facing service.

2.3.2.1 Initial design for the 5G-CLARITY slicing support system

As introduced in D2.2 the 5G-CLARITY management stratum contains a slice and service support system that is composed by the following Management Functions (MFs):

- **VIM**: Providing computer, storage and network virtualization services. Refer to Section 7.1.1 in D2.2 [2].
- **NFVO**: Providing NSD and VNF lifecycle management services. Refer to Section 7.1.1 in 5G-CLARITY D2.2 [2].
- **Transport Controller**: Providing connectivity services within the private venue. Refer to Section 7.1.2 in 5G-CLARITY D2.2 [2].
- **Multi Wat non real Time RAC**: Providing wireless service lifecycle management and configuration support services for wireless PNFs. Refer to Section 7.1.3 in 5G-CLARITY D2.2 [2].
- **Slice Manager**: Providing E2E lifecycle management services for 5G-CLARITY slices. Refer to Section 7.1.4 in 5G-CLARITY D2.2 [2].

Figure 2.9 depicts the internal architecture of the 5G-CLARITY slice support system, highlighting candidate interfaces to be used among the different management functions.

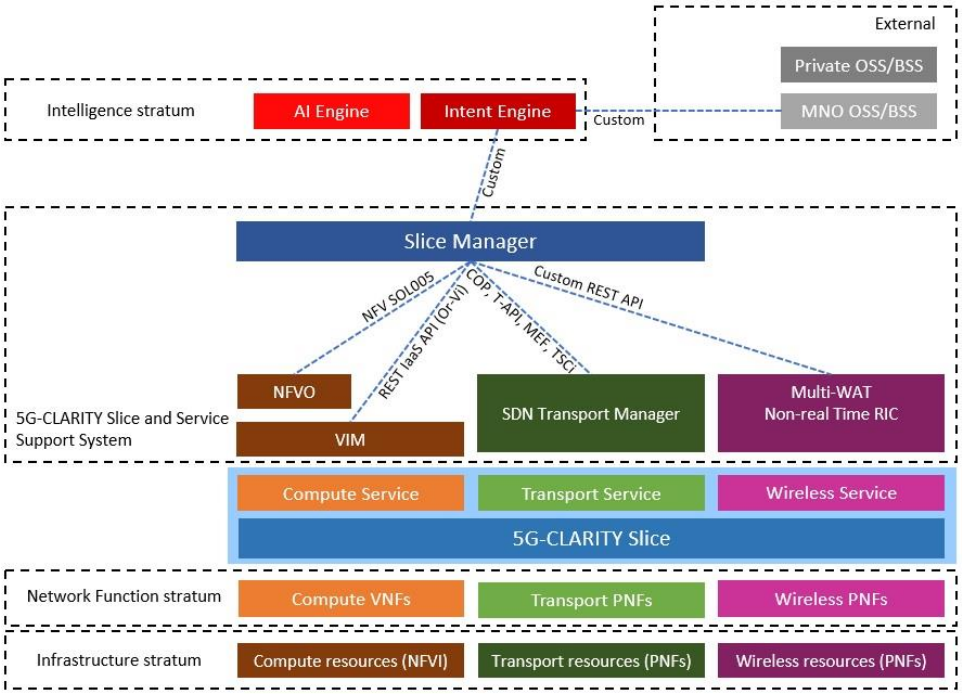


Figure 2.9. Internal architecture and interfaces of the 5G-CLARITY slice and service support system.

A major goal in WP4 will be the development of the 5G-CLARITY slice and service support subsystem, which involves the previous MFs. In order to accomplish this goal 5G-CLARITY will leverage work that is already available in the SotA, while focusing the work on the key aspects required to showcase the 5G-CLARITY concept.

The following table depicts the approach that will be followed to develop each of the MFs that constitute the 5G-CLARITY slice and service support system.

Table 2-4 Slice and Service Support System MFs.

MF	Reference Implementation	Comments
VIM	OpenStack and Kubernetes	The 5G-CLARITY slice and support system will support both VM and container-based network services.
NFVO	OSM Release 7	Provides native support for VM and container-based network services.
Transport Controller	OpenDayLight	Custom path allocation modules developed in the 5G-XHAUL and 5GPICTURE projects will be reused that expose COP interfaces
Multi-WAT non-RT Controller	Custom	Initial design developed in the 5GCITY and 5GPICTURE project will be used and extended in 5G-CLARITY
Slice Manager	Custom	An initial design developed in the 5GCITY project will be used an extended in 5G-CLARITY

Next, we describe in more detail the initial design for the multi-WAT non-RT (non-RT) Controller and the Slice Manager modules, which are the MFs that will contain the main innovations in 5G-CLARITY.

2.3.2.1.1 Initial design for the multi-WAT non real time controller MF

Figure 2.10 depicts the target architecture of the 5G-CLARITY multi-WAT non real time controller, highlighting in red new functions that will be developed or extended in 5G-CLARITY, and in orange pre-existing functions that will be extended in 5G-CLARITY.

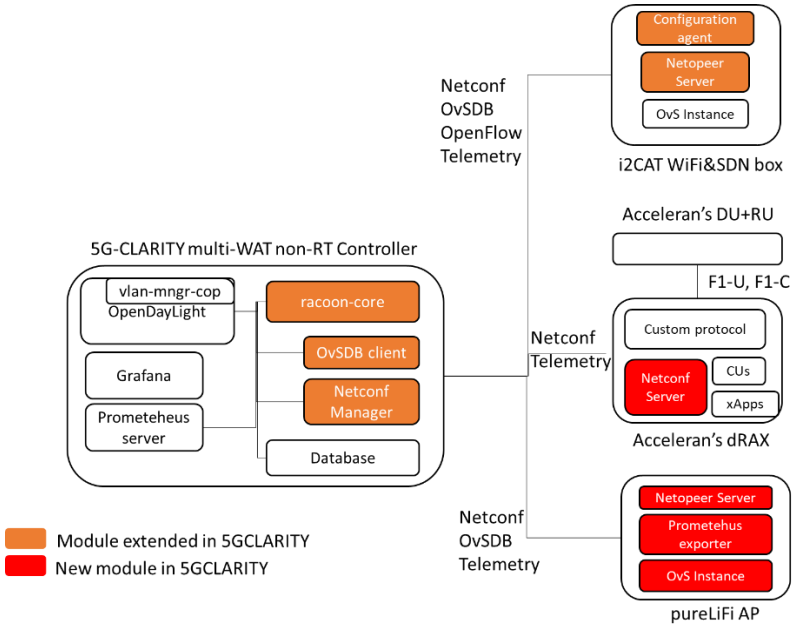


Figure 2.10. Tentative design for the 5G-CLARITY multi-WAT non-RT Controller.

The 5G-CLARITY multi-WAT non-RT controller will manage three types of physical and virtual network functions:

- The i2CAT integrated Wi-Fi and SDN switch described in 5G-CLARITY D3.1. This device will interface with the multi-WAT non-RT controller through NETCONF, OvSDB and Openflow. NETCONF will be used to provision virtual APs radiating a given SSID and to configure the wireless and wired interfaces available in the box. OvSDB will be used to provision software bridges inside the box that are used to isolate traffic from different slices (refer to 5G-CLARITY D3.1 [3]), and Openflow is going to be used to control forwarding within the integrated Wi-Fi-LiFi L2 network as described in 5G-CLARITY D3.1.
- pureLiFi's APs are controlled using NETCONF and OvSDB to provision 5G-CLARITY wireless services, as well as configuring the physical access points.
- Accelleran dRAX Virtual Network Function will provide a single NETCONF server that allows to provision PLMNIDs on the different physical small cells, as well as configuring 5GNR parameters.

The internal design of the 5G-CLARITY multi-WAT non-RT controller will follow a micro-service architecture consisting in the following modules:

- NETCONF, OvSDB and OpenFlow clients. OpenDayLight will be used as OpenFlow driver, with a custom module developed in the 5GPICTURE project (vlan-mngr-cop) that provides a higher level of abstraction and allows to provision E2E paths without the need of programming the Openflow tables in each i2CAT box.
- A core service module called 'racoon-core', which will expose all the functions of the multi-WAT non-RT controller defined in 5G-CLARITY D2.2 to the other MFs of the 5G-CLARITY management plane using a REST API. It is worth highlighting that racoon-core will expose advanced services to allow to control the amount of resources allocated in a box to a SSID or PLMNID, which will be communicated through NETCONF to the underlying box. Racoon-core will also expose a service to control the access point or small cell that a user needs to attach to. These advanced services can be exploited by the Machine Learning models in the intelligence stratum to optimize the network.

- Auxiliary modules, like a Prometheus server to gather radio telemetry from the Wi-Fi and LiFi access points, a database to maintain state that needs to persist across reboots, and a local Grafana module to enable visualization of collected telemetry.

2.3.2.1.2 Initial design for the slice manager MF

The internal architecture of slice manager of 5G-CLARITY is illustrated in Figure 2.11. The slice manager consists of two main functional blocks: 1) Network Slice Lifecycle Management, and 2) Slice Repository:

- **Network slice lifecycle management** is an entry point to the Slice Manager of the 5G-CLARITY slice and service support system that can be accessed by the Slice User via the dashboard. It is responsible for coordinating and allocating required resources and services upon request.
- **Resource manager** is an internal module of Network Slice Lifecycle Management, which is responsible for handling request for creating slices and allocate resources to the underlying infrastructures, i.e. wireless/transport/compute.
- **Service manager** is another internal module of Network Slice Lifecycle Management. It is responsible for deploying E2E network services and provision network functions, in collaboration with NFVO, for different verticals over the allocated slices.
- **Slice repository** is responsible for storing information related to the network slices. It stores Information related to the virtual and physical resources of the slices, such as which virtual resources belong to the slice, which physical resources they are mapped to, and to whom each slice belongs.

The slice manager also includes a **northbound API** and a set of Endpoints. The northbound API is a subcomponent that is envisioned to provide a REST interface to Slice Manager, enabling the access to the slicing functionalities to authorized parties. Northbound API will also provide a complete overview of Slice Manager features and information. To achieve so, it will interact with different Slice Manager's components in order to retrieve the requested information correlating multiple sources of data, if needed. The endpoints are used to provide interface to the underlying resources, including NFVO, VIM, multi-WAT non-RT controller and transport controller.

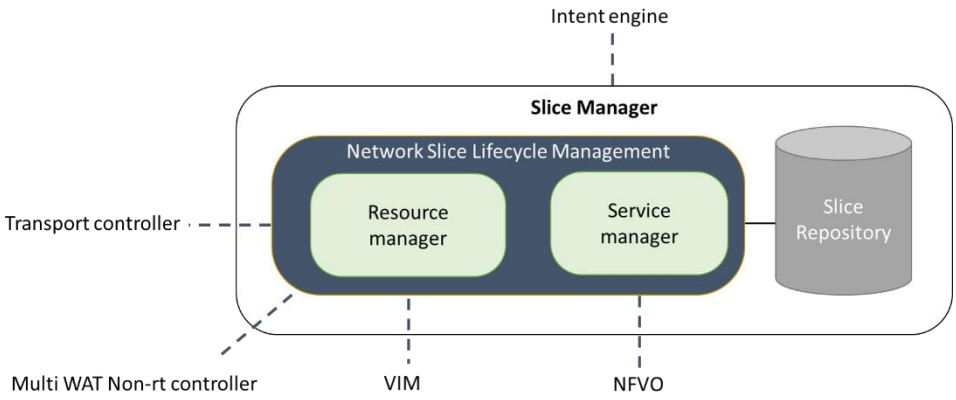


Figure 2.11 Tentative design for slice manager MF.

2.3.2.1.3 Example 5G-CLARITY slice and service blueprint and provisioning flow

To conclude our initial design of the 5G-CLARITY service and slice support system we provide in this section a detailed description of how a 5G-CLARITY slice could look like, and we provide a preliminary signalling flow illustrating the various phases of the slice provisioning workflow.

Figure 2.12 illustrates an example 5G-CLARITY infrastructure, hosting two 5G-CLARITY slices, along with their associated services. For each slice, the figure highlights the wireless, transport and compute services that

compose the slice.

Starting with the first slice, called "5GCL slice 1" in the legend box in the figure, we see it is composed of:

- **Wireless services**: There are three wireless services that are part of this slice. A cellular wireless service in the gNB RU+DU radiating PLMNID-XYZ, which in addition includes two 3GPP slices identified as S-NSSAI1 and S-NSSAI2. In addition, we have two wireless Wi-Fi services identified with SSID-1 and SSID-2, which will connect to the respective S-NSSAIs. Wireless quotas for these services can be defined in the gNB for the PLMNID+S-NSSAI pairs, and in the Wi-Fi APs on a per-SSID basis.
- **Compute services**: We see three ETSI network services being part of this slice. NS0 contains the control plane functions of the 5GCore and the N3IWF function required to process the traffic coming from the Wi-Fi APs. NS1 contains a UPF and an application serving the traffic connected to S-NSSAI1, and NS2 contains a separate UPF and another application serving the traffic connected to S-NSSAI2. Three separate compute chunks have been set up to guarantee compute resources for each of these network services.
- **Transport Services**: There is a long list of transport services, identified with a VLAN, required to implement the connectivity between the wireless and the compute services of this slice. First, we see **vlan11** that is used to connect the control plane traffic from the RAN cluster to the 5GC control plane components in NS0. Also, from the RAN cluster we see vlan21 and vlan31 that carry user plane traffic towards the UPFs belonging to each S-NSSAI connecting with NS1 and NS2. From the Wi-Fi AP we see two additional VLANs, namely vlan20 and vlan30, which connect the traffic from SSID-1 and SSID-2 to the N3IWF function in NS0, from which traffic is then diverted to the appropriate network service NS1 or NS2. Finally, there are two additional transport services vlan22 and vlan32 used to differentiate the traffic destinated to S-NSSAI1 and S-NSSAI2 in the mid-haul segment. Notice that in the mid-haul segment user plane traffic is encapsulated in a GTP tunnel between the gNB-DU and the gNB-CU in the F1 interface. In this interface traffic from individual users can be distinguished looking at the GTP Tunnel Endpoint Identifier (TEID) field. Thus, a software bridge with GTP capabilities could be used in the gNB-DU to push the correct VLAN based on the GTP TEID field.

A similar analysis can be carried out for slice 2, called "5GCL slice 2" in the legend box in Figure 2.12 to identify its components wireless, compute and transport services. In this case this slice only has one S-NSSAI and one SSID radiated in the venue, and two component network services sharing resources inside the same compute chunk. Unlike the first slice, the second one does not implement a full 5GC, but instead it relies on an MNO to provide the full 5GC functions. Inside the venue the second slice only features a first network service with an N3IWF used to process the Wi-Fi traffic and a virtual gateway function to connect to the MNO where the 5GC control plane resides, and a second network function featuring the UPF and an application function used to process the traffic.

As we can see from the provided example a 5G-CLARITY slice and the services that the slice supports require a complex configuration of physical network functions (wireless and transport) and virtualized network services. Figure 2.13 provides a tentative signalling flow illustrating how slices can be provisioned by the MFs in the 5G-CLARITY slice and service support system.

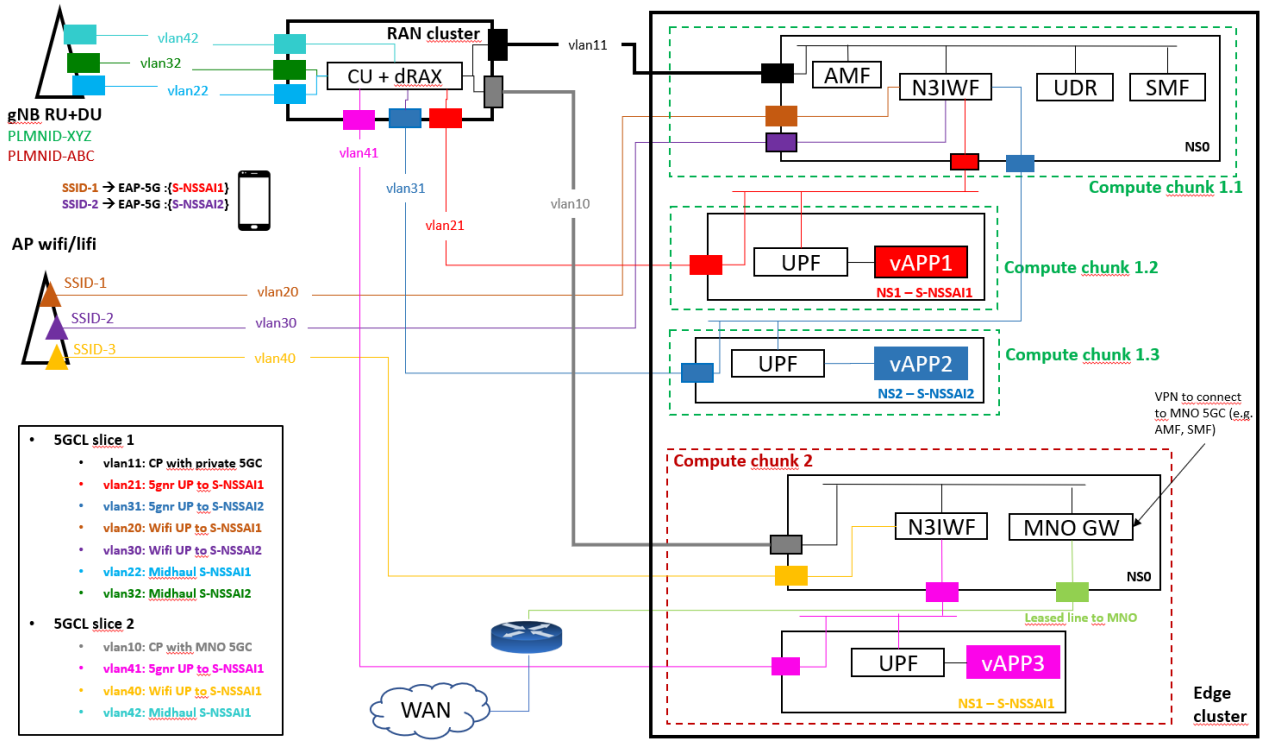


Figure 2.12 Example of a 5G-CLARITY deployment containing one 5G-CLARITY slice with 2 NSSIs for the private operator and one 5G-CLARITY slice with one NSSI from an MNO.

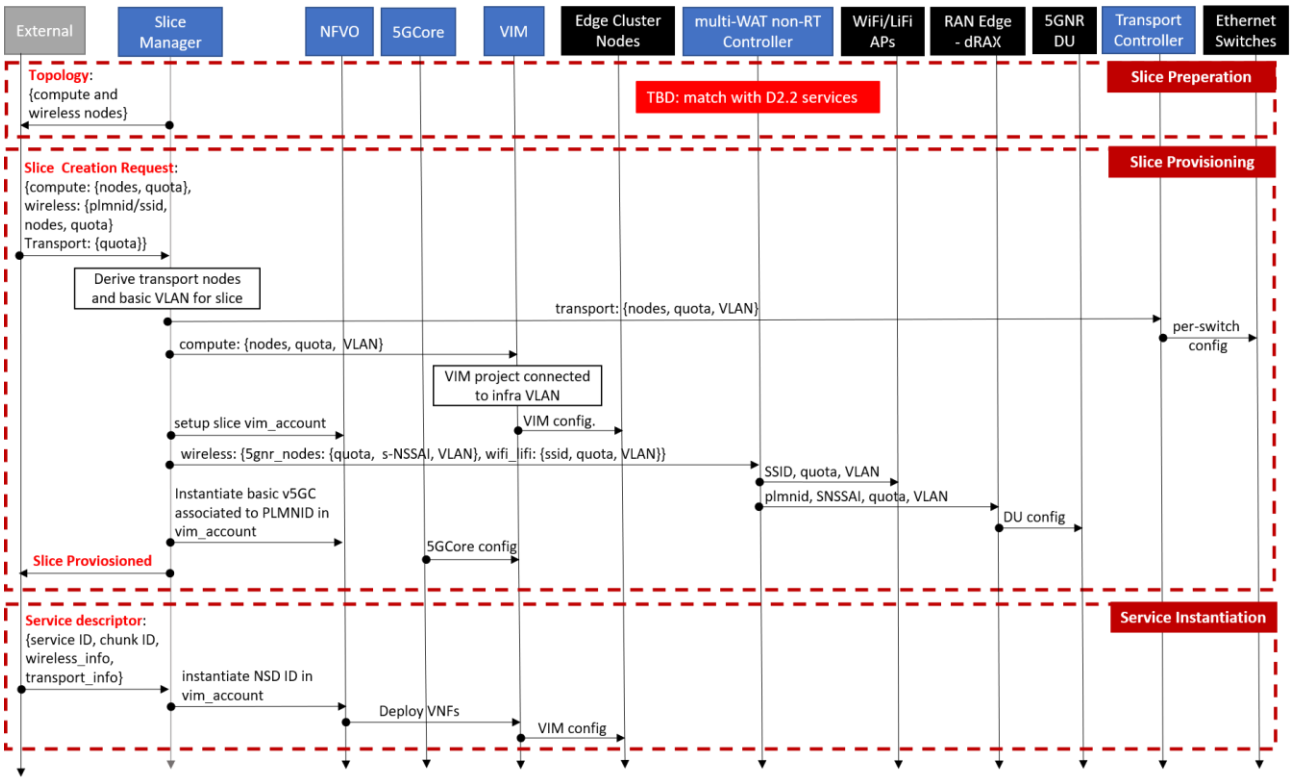


Figure 2.13 Tentative message flow to provision a 5G-CLARITY slice.

To provide a 5G-CLARITY slice, several phases must be performed by the 5G-CLARITY management architecture. These phases are depicted in Figure 2.13 and defined below:

- Phase 1. Preparation phase: occurs before a 5G-CLARITY consumer (i.e., the private owner, the public

network or a third party) requests a 5G-CLARITY slice. This phase includes the registration of infrastructure resources in 5G-CLARITY platform.

- Phase 2. Instantiation and Configuration Phase: starts when a 5G-CLARITY consumer requires a 5G-CLARITY slice with specific requirements. This phase comprises of, *a)* browse the available infrastructure and select the nodes from the 5G-CLARITY dashboard; *b)* check the deployment feasibility of the 5G-CLARITY slice; *c)* create chunks of components and assigns quota to each created chunk, including the onboard of the correspond NSs; *d)* the configuration at application level of the NFs (i.e., those implemented as VNFs within the deployed NSs and the APs implemented as PNFs) to provide the 5G-CLARITY slice the desired configuration; and *e)* finally, the 5G-CLARITY slice is activated. This means the 5G-CLARITY slice is ready to be consumed.
- Phase 3. Provisioning of NSs: network services and application related services can be deployed over the activated slice.

Focusing on Phase 1, the entities involved are the slice manager and external entity. The role of external entity could be taken by the private owner (i.e., slice as private network operator internals), by the public operator, or by a third party (i.e., network slice as a service). The first step is that the slice manager registers the infrastructures through the 5G-CLARITY platform, and the second step is to make them available for the external entity.

In Phase 2, the external entity requests the specific 5G-CLARITY slice. In this step Slice Manager derive the transport node required to interconnect the 5G-CLARITY compute services with the 5GNR/Wi-Fi/LiFi APs providing the wireless services. In this procedure, the VLAN tags used in the 5G-CLARITY transport service are created. After this procedure, the Slice Manager communicates these parameters to the transport network Manager, which is responsible for properly configure the Ethernet switching devices. At this point, the 5G-CLARITY transport service is created. After that, the Slice Manager must communicate with the VIM to reconfigure the access points of each 5G-CLARITY compute service according to the created 5G-CLARITY transport service. When the transport and compute services of the 5G-CLARITY slice are created, the NFs (i.e., implemented as VNFs or PNFs) must be configured at application level. In the case of the constituents VNFs of each 5G-CLARITY compute service, software-related parameters will be configured. In the case of 5GNR/Wi-Fi/LiFi used in each 5G-CLARITY wireless service, the 5G-CLARITY wireless quotas as well as the supported S-NSSAIs and PLMN IDs/SSIDs will be configured. Finally, the Slice Manager instantiates and configures v5GCore associated to the PLMNID in the VIM. At this point, the 5G-CLARITY slice is ready to be operating. Therefore, the Slice Manager notifies the external entity (i.e., in turns, it will forward it to the network slice consumer) that the 5G-CLARITY slice is active.

Finally, in Phase 3, the service descriptor should be Instantiated in the NFVO by the Slice Manager. In this case, the NS descriptor (NSD) will be used by the NFVO to deploy the network service. During this procedure, the NFVO interacts with the VIM to allocate the virtualized compute, networking and storage resources required by the NS instance.

2.4 Multi-domain telemetry

2.4.1 State of the art

ETSI ISG ENI defines how AI can be usefully applied in telecommunication networks to support the management objectives of operators [67]. These include making management faster, more efficient and providing higher resilience and reliability of the infrastructure and of the services delivered to end-users. ETSI ENI considers number of technical factors that are needed to be taken into account to determine the degree of AI based autonomy in a network and one of these technical factors is network telemetry. Following table describes levels of device awareness and examples of telemetry being collected.

Table 2-5: Data Collection and Awareness Parameters

Level	Device Awareness	Example Telemetry
Level 1	Single device and shallow awareness	SNMP events and alarms are collected
Level 2	Local awareness	SNMP events, alarms, KPIs, and logs are collected
Level 3	Comprehensive awareness	Telemetry data is collected
Level 4	Comprehensive and adaptive sensing	Data collected is compatible with data compression and optimization technologies
Level 5	Data-based adaptive posture awareness	Edge data collection and judgment based on data
Level 6	Data-based adaptive optimization upon deterioration	Edge closed-loop processing, including collection, judgment, and optimization

To increase the level of autonomy in the networks, telemetry frameworks are required to evolve and to support complex multi-domain scenarios. This section provides a view of what components are needed to be able to support multi-domain AI based optimizations in the network. In this context:

- **non-RT (non-RT) telemetry** includes collection and processing of measurements and data from network and end-user equipment that has no immediate consumers. Non-RT telemetry may include system state, logs, configurations, or other text / binary data. Some of the telemetry may be unstructured and stored as it has been received and other telemetry may be strictly based on standard models.
- **Real-time telemetry** means collection of measurements or other data in real-time at remote nodes and automatic transmission of the data. The nodes are transmitting real-time telemetry to peer nodes deployed either at the edge or at cloud. Real-time telemetry may be consumed also by non-RT functions, for instance in a data lake scenario.
- **Multi-domain telemetry** consists of non-real time and real-time telemetry that is collected from end user devices or network devices across multiple network domains. Table 2-6: Multi-Domain Telemetry Categories categorizes the sources of multi-domain telemetry.

Table 2-7 surveys cloud native components for telemetry available in the SotA that can be used to address the telemetry categories introduced above.

Table 2-6: Multi-Domain Telemetry Categories

Category	Examples
Measurements	Measurements originating from UEs, network statistics, RAN measurements (real-time or logged).
Configurations	Real-time or non-RT configurations per each infrastructure or system node. Configurations may be either structured or unstructured (e.g. when stored to data lake).
Logs	State transformations of a node or an application. Errors, Network Node configurations, Orchestration configurations from single nodes. Logs may be either structured or unstructured.
Tracing	Tracks configuration changes across multiple infrastructure / VNF nodes. Different from logs, traces provide more detailed information to debug the issues/problems and can be collected via pub-sub system where data consumers are collecting a trace from data pipeline.
System Monitoring Telemetry	Infrastructure as well as network functions are monitored to generate alerts when the system is not operating within the limits it has been set.
Telemetry Pipeline	Component of the telemetry system that transmits multi-domain stream or batch telemetry.

Table 2-7: Cloud Native Products for Telemetry Collection

Product	Category	Features
Prometheus	Monitoring	Prometheus's main features are: • a multi-dimensional data model with time series data identified by metric name and key/value pairs • PromQL, a flexible query language to leverage this dimensionality • no reliance on distributed storage; single server nodes are autonomous • time series collection happens via a pull model over HTTP • pushing time series is supported via an intermediary gateway • targets are discovered via service discovery or static configuration • multiple modes of graphing and dashboarding support • federation of Prometheus servers
Apache Kafka	Telemetry Pipeline	Kafka provides three main functions • Publish and subscribe to streams of records • Effectively store streams of records in the order in which records were generated • Process streams of records in real time
Grafana	Monitoring	Grafana allows you to query, visualize, alert and understand your metrics no matter where they are stored. Create, explore, and share dashboards with your team and foster a data driven culture. Main features: Visualization, Dynamic Dashboards, Ad-hoc log queries, log search, log alerts, GUI for rules and data source plugins.
Fluentd	Logging	Fluentd is an open source data collector, which lets you unify the data collection and consumption for a better use and understanding of data. Fluentd supports large number of off-the-shelf plugins for variety of systems.
Jaeger	Tracing	Jaeger, is a distributed tracing system. It is used for monitoring and troubleshooting microservices-based distributed systems, including: • Distributed context propagation • Distributed transaction monitoring • Root cause analysis • Service dependency analysis • Performance / latency optimization
AWS	Cloud Platform	AWS is a cloud computing platform provided by Amazon. It comprises an excess of 175 services, which includes computing, networking, storage, database, analytics, IoT. Additionally, AWS services can be accessed using APIs, such as HTTP using the RESTful APIs.

Next, we provide a SotA review on two telemetry components that are key to the 5G-CLARITY system, namely data lake technologies and streaming telemetry.

2.4.1.1 Data lake technologies

A data lake is an architectural pattern of large data repository that allows storing of structured and unstructured data and is scalable based on user needs. The data can be stored based on organizational needs. Data Lake combines a large-scale storage repository with a variety of high-performance processing engines that are usually virtualized. Large scale and independent scalability of processing and storage allows data lakes to store data as it is produced without pre-processing and processed it on-access. This approach allows flexible usage of produced data and it enables organizations to run different types of analytics from dashboards and visualizations to big data processing, real-time analytics, and machine learning. Some

functions of data lakes are:

- Data Lakes allow data-based access controls which enables multiple roles within the same organization or external organizations to gain access to a specific data for a specific time. For instance, in one organization data scientists, data developers, and business analysts can access the same underlying data with their preferred tools and frameworks. This includes open source frameworks such as Apache Hadoop [68], Presto [69], and Apache Spark [70], and commercial products from data warehouse and business intelligence vendors.
- Data Lakes allow to run analytics without the need to move data to a separate analytics system.
- Data lakes allow large data storages to be tiered based on access frequency. It is possible to implement data access strategies based on tiering for efficiently using storage technologies according to the business value or importance of the data. A tiered storage system provides several types of storage, for example, SSD disk drives, HDDs and tape storages. In some cases, the tiering can be automated based on last data access information and tiering strategies may be based on, availability, and performance of the object storage. It is possible to configure storage classes configured object level and a single bucket can contain objects stored across data lake.

2.4.1.2 SNMP and streaming telemetry

The demand for data regarding network state, whether to detect hot spots in the network, or to aid decision making on workload placement requires data at a rate that traditional methods cannot deliver. Traditional methods of collecting network telemetry data include *pull*-based mechanisms such as the Command-Line Interface (CLI) shows commands, Syslog messages, IP Flow Information Export (IPFIX) notifications, and more recently, Simple Network Management Protocol (SNMP) [71]. SNMP protocol addresses the limitations of CLI, Syslog and IPFIX protocols (see Figure 2.14 CLI, Syslog and IPFIX.), allowing the collection and aggregation of different performance metrics (e.g. availability, throughput, utilization, delay, fault notifications) from a wide variety of devices, based on the simple installation of appropriate SNMP agents atop them. This feature makes SNMP ideal for use in heterogeneous networks, typically consisting of devices deployed across different technology domains and provided by different vendors. Indeed, SNMP has replaced CLI, Syslog and IPFIX in today's commercial TCP/IP networks, becoming the de-facto solution for data collection and aggregation in multi-technology, multi-vendor network environments.

Figure 2.14 CLI, Syslog and IPFIX.

Every network management system primarily uses today SNMPv3. This protocol version leverages on traditional SNMP model, which defines two entities that work in a client-server mode:

- **The SNMP agent** (i.e. SNMP client)**,** located on every device that needs to be monitored. The software implementing the SNMP agent consists of three main artifacts: *i)* the SNMP transport protocol stack, used for client-server communication; *ii)* the SNMP agent engine, processing client requests and formatting data; and *iii)* the agent profile, providing rules that control access to

Management Information Base (MIB) [72] variables and manage which client requests are authorized.

- **The SNMP manager** (i.e. SNMP server), installed in the operator's network management system. The SNMP manager job is to acquire information from the different devices connected to the network, collecting data from their individual SNMP agents. These data include *performance metrics*, exchanged between the SNMP manager and the SNMP agents using SNMP request-response messages, and *event notifications*, sent from individual SNMP agents to the SNMP server in form of SNMP traps. These traps are asynchronous, unacknowledged alert messages that are used to inform the SNMP manager when an important event (e.g. failure) happens at the SNMP agent level. The SNMP manager software includes, *i)* a database, used to store collected data; *ii)* the SNMP transport protocol stack; *iii)* the SNMP server engine, which is the kernel of the SNMP, managing all the tasks like an orchestral chief; and *iv)* a set of agent management profiles, providing rules that define how to access to the different agents, and that helps the SNMP manager to build the topology map.

The ways of working in SNMP are illustrated in Figure 2.15. The SNMP agent listens to request coming from the SNMP manager on the UDP port 161, while the SNMP manager listens to alarms "TRAP" coming from the agent on UDP port 162.

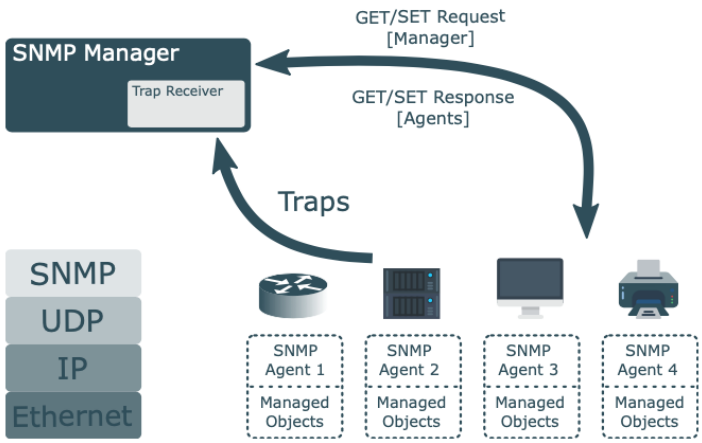


Figure 2.15 SNMP protocol.

Despite the wide adoption of SNMPv3, this solution has some limitations that are inherent to the polling-based techniques. Some of these are described below:

- **Lack of automation**. Under the poll-based mechanism, the devices send data only when requested by a client. In SNMP, this requires selecting SNMP agents and metrics to poll, setting polling intervals, etc. All of these actions require human intervention.
- **Slow reaction capabilities**. When using SNMP, for instance, changes that occur between a polling interval are discovered only after the next polling. SNMP polling intervals are typically in the order of 5-10 minutes, which is unacceptable in operational networks when thousands of events can occur in the interim.
- **Lack of useful data**. The SNMP protocol does not allow collecting all relevant information of every device, preventing the SNMP manager to have a full picture of the network status. This is because some device information is not stored in the MIB, and therefore it is accessible only using CLI command. Additionally, SNMP protocol does not allow collecting historical routing information either, thus leaving a gap into routing changes in the network.

- **Strict semantics**. SNMP has tight ordering and very limited extensibility.
- **Scalability burdens**. In large-scale networks, the sheer number of devices the SNMP manager has to poll and the volume of data it has to process causes overloads in the operator's network management system.

The above limitations make SNMP (and in general any poll-based mechanism) inappropriate for beyond 5G networks, where operators require near-RT access to operational statistics from a wide variety of devices is required. Thus, SNMP-based network monitoring is long overdue for an upgrade, requiring operator to find alternative solutions. In this regard, a new paradigm has emerged, i.e., streaming telemetry.

Streaming telemetry represents a new approach for network monitoring whereby the required data can be 'streamed' automatically and continuously from various network devices, without the need for any polling. Streaming telemetry builds on two main pillars:

- near-RT network data achieved with **push-based** data collection. Unlike polling, this mechanism allows pushing data from the device to an external collector at a much higher frequency and more efficiently. This collector is integrated in the operator's network management system.
- The use of **data model** to configure and manage individual network devices in a programmatic way. With this model, network devices can be configured with the type of data to be collected, the frequency of collection, and where it should be sent. Telemetry data is typically described using YANG [73], a structure data modelling language, encoded in JSON, XML or Google Protocol Buffers (GPB), and streamed over NETCONF [74], RESTCONF [75] or gRPC [76]. For more information, see Figure 2.16.

The basic operation of streaming telemetry is shown in Figure 2.17. In order to stream data from a given device, the collector must set up a 'subscription' to a data set in a YANG model. A subscription represents a contract between a subscription service and a subscriber that specifies the type of data (i.e. data items) to be pushed. The subscription allows the collector (i.e. subscriber) to subscribe to data models and device (i.e. publisher) to push the data to the collector for the subscribed model.

With this mechanisms, streaming telemetry enables access to real-time, model-driven, and analytics-ready data that can help with network automation, traffic optimization and preventive troubleshooting in large-scale networks. To allow operators to fully exploit these features into their managed networks, typically consisting of multiple devices using multiple technologies and provided by different providers, telemetry streaming needs to be accompanied with a highly scalable architectural frameworks, with more data point granularity and superior performance. Examples of these frameworks are shown in Figure 2.18. On the left side (Figure 2.18a), the ETSI ISG Context Information Management (CIM) architecture is represented. This architecture allows addressing what today is a hindrance to the widespread adoption of massive IoT services, i.e., the lack of open and standardized approach for the exchange of context information. To this end, ETSI ISG CIM defines an open framework based on the use of RESTful APIs named NGSI-LD for a consistent, cross-cutting context exchange [77]. For more information on NGSI-LD protocol and how streaming telemetry can be applied therein, see [78]. On the right side (Figure 2.18b), there is the IETF-defined Service Assurance for Intent-based Networking (SAIN) architecture [79]. This architecture makes use of telemetry data streamed from multiple devices to get the assurance of healthy services running atop.

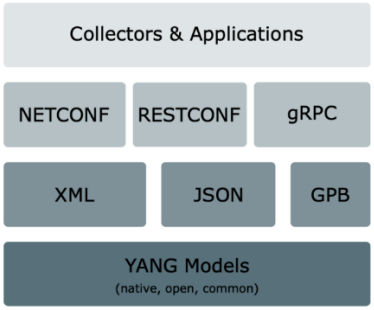


Figure 2.16 Telemetry data protocol stack.

Figure 2.17 Basic operation of streaming telemetry.

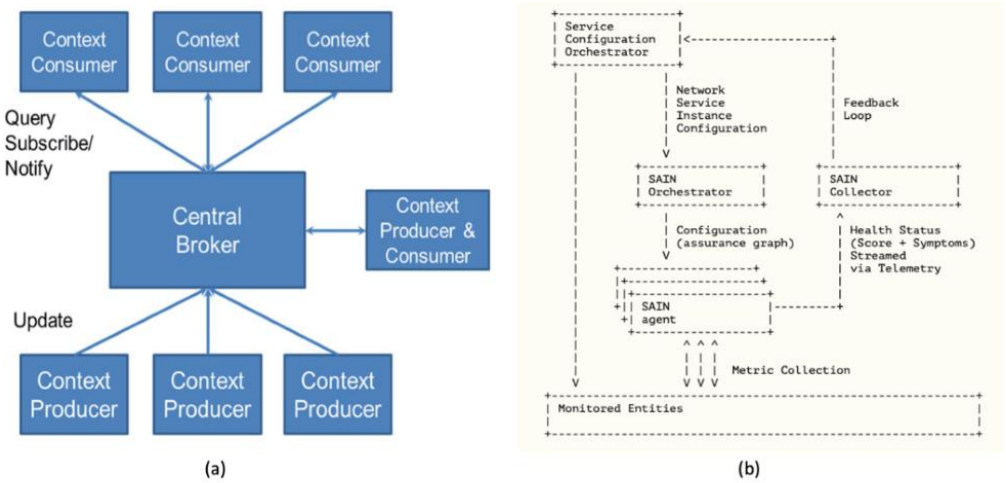


Figure 2.18 Examples of architectures for streaming telemetry applicability.

2.4.2 Initial design of 5G-CLARITY data processing and management subsystem

Based on the SotA technologies described in the previous section, now we present an initial design of the 5G-CLARITY Data Management and Processing subsystem that will be developed in WP4. Figure 2.19 describes the overall architecture of this 5G-CLARITY subsystem, including two main components, *i)* the Data Semantic Fabric; and *ii)* the Data Lake.

The Data Semantic Fabric provides a model-based telemetry framework for data aggregation in distributed, multi-domain environments. This framework leverages two key functionalities. On the one hand, the ability to apply a semantic model to telemetry data collected from multiple sources. This model allows providing a complete description of data flows, including the identification of data sources and data consumers, and

their relationships with the rest of elements in the flow. On the one hand, the ability to combine individual data and process them according to predefined rules, in order to make data available for consumption. The focus of the Data Lake is to centralize the data management and allow virtual unlimited data storage allowing a cost-effective management of data and its access.

As shown in Figure 2.19, the Data Semantics Fabric communicates with the Data Lake for the purposes of **data storage** and **data collection**. In the first case, the Data Semantic Fabric dispatches aggregated data to the Data Lake's storage system, from which it can be read later on by corresponding data consumers. Service and slice provisioning MFs (5G-CLARITY management and orchestration stratum) and AI engine (5G-CLARITY intelligence stratum) are examples of clients that may gain access to the Data Lake to consume the stored aggregated data. In the second case, the Data Semantic Fabric uses the Data Lake as another data source. By modelling Data Lake's storage system as a database, the Data Semantic Fabric can fetch necessary data, using it for aggregation with other data sources.

Figure 2.19 also highlights the multi-domain nature of the 5G-CLARITY system, where data may originate from different sources, including WAT nodes (5G-CLARITY infrastructure stratum), compute and transport nodes (5G-CLARITY infrastructure stratum) and VxFs (5G-CLARITY network and application function stratum).

Detailed description of the internal components 5G-CLARITY Data Processing and Management subsystem are presented in the following sections, i.e., leveraging the ability to obtain real-time multi-WAT telemetry data in Section 2.4.2.1, the Data Semantic Fabric in Section 2.4.2.2 and the Data Lake in Section 2.4.2.3.

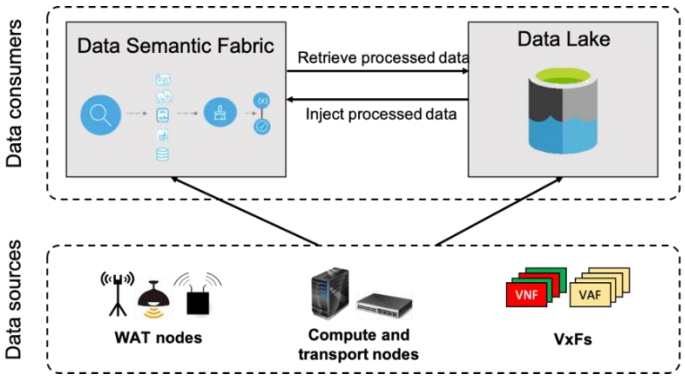


Figure 2.19: Data processing and management subsystem.

2.4.2.1 5G-CLARITY approach to collect multi-WAT telemetry

A central aspect to fulfil the 5G-CLARITY vision is to be able to extract near real time metrics about the wireless access technologies that compose the 5G-CLARITY network, namely 5GNR, Wi-Fi and LiFi. These metrics are essential for example for the operation of some of the ML based algorithms introduced in Section 5 Obtaining RAN information is however a complex task in real networks because interfaces offered by equipment vendors are generally proprietary. To address this problem, in 5G-CLARITY we make use of the near-RT RAN Intelligent Controller (RIC) concept proposed by O-RAN, which has been introduced in 5G-CLARITY D3.1 [3]. O-RAN defines an interface called E2 between the RT RIC and the Central Unit (CU) functions, which allows to retrieve radio information from the UEs connected to the gNB. The RIC implementation in 5G-CLARITY will be based on the dRAX product from ACC, which offers a data bus based on a pseudo-E2 interface that allows different xApps to publish and subscribe to relevant information. dRAX will enable the easy lifecycle management and access to telemetry of the xApps developed in 5G-CLARITY. In this sense a "multi-WAT KPI xApp" will be developed that collects measurements on a per-slice basis and delivers them to the DSE component of the overall 5G-CLARITY telemetry system. A similar "enhanced AT3S" xApp will be developed as described in 5G-CLARITY D3.1 [3] that will use multi-WAT telemetry data to control

the weights and policies of the different multi-WAT flows.

One aspect that is not covered in O-RAN is how to integrate telemetry from non 3GPP technologies. In this regard, 5G-CLARITY will demonstrate a novel solution that will use Prometheus to collect Wi-Fi and LiFi telemetry and then an adapter to publish the metrics into the dRAX data bus. In this way both the 3GPP and non-3GPP radio metrics will become available to the multi-WAT KPI xAPP. Figure 2.20 depicts the proposed multi-WAT telemetry architecture, which will be hosted in the RAN compute cluster described in 5G-CLARITY D2.2 [2].

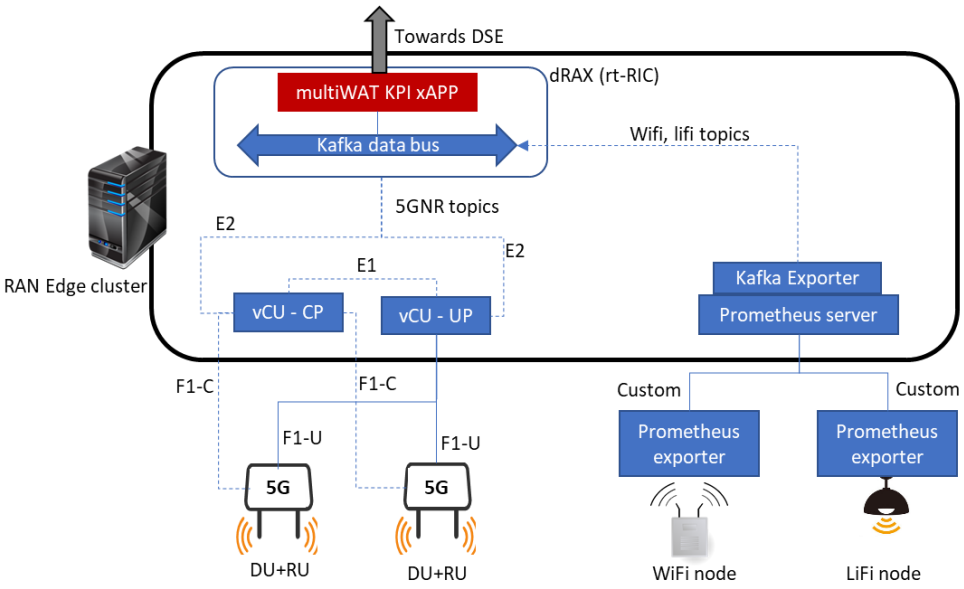


Figure 2.20. 5G-CLARITY approach to multi-WAT telemetry.

2.4.2.2 5G-CLARITY data semantic fabric

The streaming telemetry principles described in Section 2.4.2.1 will be applied in the design of the data semantics fabric. This fabric, which builds up the 5G-CLARITY data pipeline, allows consolidating data from a wide variety 'sources' and turn them into useful information for 'consumers', by applying necessary processing on the collected data before their transmission and storage. For an effective data ingestion, the 5G-CLARITY data pipeline includes a set of logical nodes depicted in Figure 2.21, each with a well-defined functionality:

- **Collector,** responsible for data harvesting.
- **Aggregator,** in charge of manipulating and combining individual data collected together, making them available for their consumption. This processing is done according to some rules (e.g. arithmetic operations, filtering, thresholding).
- **Dispatcher,** which sends aggregated data out to the target destination. This destination is where the data is stored for their consumption.

Given the above, we can see that streaming telemetry applies to the collector. Examples of sources from which the collector can retrieve data include infrastructure equipment (e.g. network devices, compute nodes), instances of manageable network entities (e.g. network slices/services, VNFs, probes) and databases. Every data source has associated a well-defined class, which provides a complete description of this source. A class consists of two types of information:

- Endpoint of the source, including the URI (e.g. path, host, port) and the corresponding credentials (e.g. user, password, method). This information allows the collector to establish a subscription with

the source.

- The data items (e.g. in-octets, in-pks, in-unicast-pkts, MTU size) that can be retrieved from this source by a subscribed collector. These data items, which convey the semantics of the data source, are typically structured into a standards-based YANG model. YANG is a data modelling language based on the structure of management information, next-generation system, being used to model semantics and organization of configuration and state data manipulated by the NETCONF protocol. To retrieve data from a source, the collector subscribes to specific data items it needs, by using the YANG model embedded in the class. Figure 2.22 illustrates the class of a typical YANG-based network device.

Finally, an initial set of telemetry data sources to be integrated with the Distributed Streaming Telemetry Pipeline have been already identified and are listed in Table 2-8.

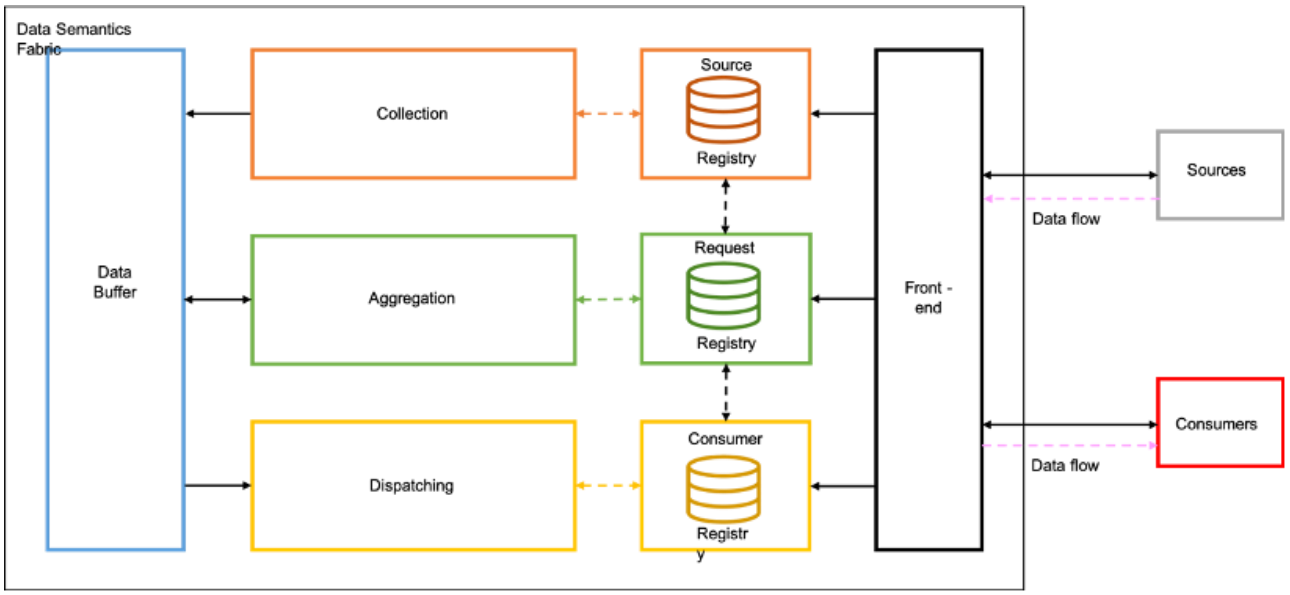


Figure 2.21: Data semantic fabric.

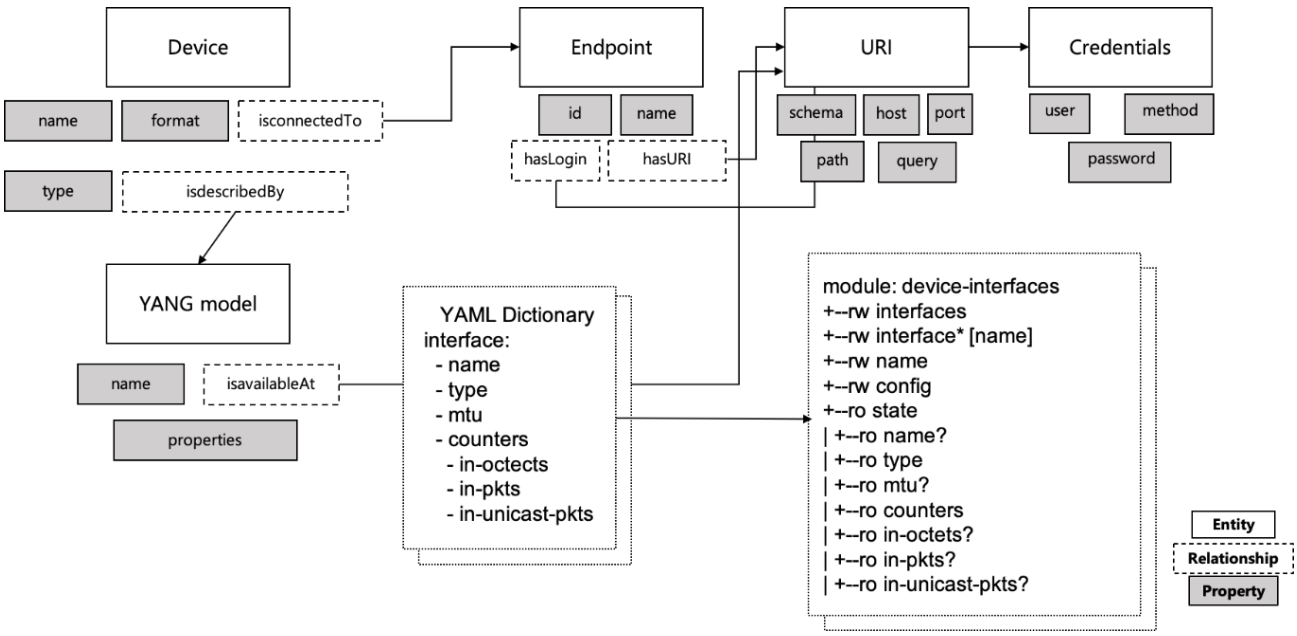


Figure 2.22: Information model of a YANG-based device.

Table 2-8: Telemetry Data and Corresponding Data Sources in 5G-CLARITY

Telemetry Data	Data Source	Comments
multi-WAT telemetry including 5GNR, Wi-Fi and LiFi	dRAX xAPP described in Section 2.4.2.1	Open source telemetry framework will be used (e.g. Prometheus)
L2-based network telemetry	SDN controller	Through controller NBI offered APIs
	Individual network devices	Devices with in-built NETCONF server
NFVI telemetry	OSM mon-collector	Collects data from VIM's telemetry system, e.g. OpenStack Ceilometer
VNF telemetry	OSM mon-collector	Collects data from OSM's N2VC, based on Juju
E2E measurements	AT3S	In case MP-TCP is used to implement the AT3S user plane function, then MPTCP E2E performance metrics are available [80]
5GC analytics	NEF	Secure exposure of NWDAF provided data
CPE telemetry	CPE device	

2.4.2.3 5G-CLARITY cloud-based solution for multi-WAT telemetry

Analysing large data comes with a number of challenges, which include infrastructure, cost, storage and security. One solution to these challenges relies on cloud computing, which migrate the in-house infrastructure requirement to an external platform. In this section, we propose a cloud-based approach for handling multi-WAT telemetry, which is based on the AWS platform. This approach is complementary to the solutions presented in Sections 2.4.2.1 and 2.4.2.2.

AWS is a cloud computing platform provided by Amazon. It comprises a multitude of services, which includes computing, networking, storage, database, analytics and IoT. The bulk of AWS services lie in the background and are not exposed to the consumer, they can avail these services only through API calls.

Figure 2.23 illustrates the overall framework of the cloud-based solution targeted in 5G-CLARITY. The framework consists of two logical layers, namely the network layer and the computing layer. The computing layer is composed of different computing tiers, namely, the central cloud and the edge cloud connecting to the RAN or core, while the network layer is depicted using and E2E multi-WAT network.

This solution extends the telemetry handling to the cloud side via an edge premises. To expound further, the edge shown in Figure 2.23 forwards the incoming telemetry from the available wireless access technologies, such as Wi-Fi, LiFi and 4G/5G, towards the AWS cloud. In one exemplary scenario, this may be carried out using the Greengrass service, which is an AWS service that extends AWS to edge devices so they can act locally on the data they generate and use the cloud for management, analytics and durable storage.

The incoming telemetry is monitored both at the edge and the cloud side using AWS Rules. The AWS Rules incorporates a set of predefined and customizable rules, which allows triggering some functions defined in the Lambda Functions service. For example, a rule can be defined to trigger a Lambda function that forwards an incoming telemetry from, say, a Wi-Fi access technology to a specific AWS service.

Additionally, the incoming telemetry data stream can be connected to a Kinesis Data Firehose service, which is a data streaming service equivalent to the Kafka service employed in Section 2.4.2.1. The Kinesis Data Firehose service loads streaming telemetry data into a storage space, particularly the Simple Storage Service (S3) storage service shown in Figure 2.23. The S3 storage service offers industry-leading scalability, data availability, security, and performance. In the context of 5G-CLARITY, this would be the main storage bucket for all the received telemetry. This storage can be also accessed by other tools, such as analytics tools and

ML-based applications.

Telemetry is typically communicated using the JSON format. One JSON-based format referred to as the Signal Metadata Format (SigMF) specifies a way to describe a real or complex time series signal, with one or more non-continuous captures, using metadata written in SigMF compliant JSON. SigMF can be used to describe general information about the equipment and method used for collection of samples, the characteristics and parameters of the system that recorded the samples, and features of the signal itself. It is intended to be generally applicable to signal processing, regardless of whether or not the application is communications related with the goal of providing a standard for time-series data that will be useful regardless of tool or workflow.

The SigMF specification fundamentally describes two types of information: datasets, and metadata associated with those datasets. Taken together, a dataset with its SigMF metadata is a SigMF Recording. Metadata describes the dataset with which it is associated. There is a one-to-one mapping between SigMF data files and SigMF metadata file. The metadata includes information meant for the human users of the dataset, such as a title and description, and information meant for computer applications (tools) that operate on the dataset. A SigMF Recording consists of two files, i.e., a SigMF metadata file; and a dataset file. The dataset file is a binary file of digital samples, and the metadata file contains information that describes the corresponding dataset file.

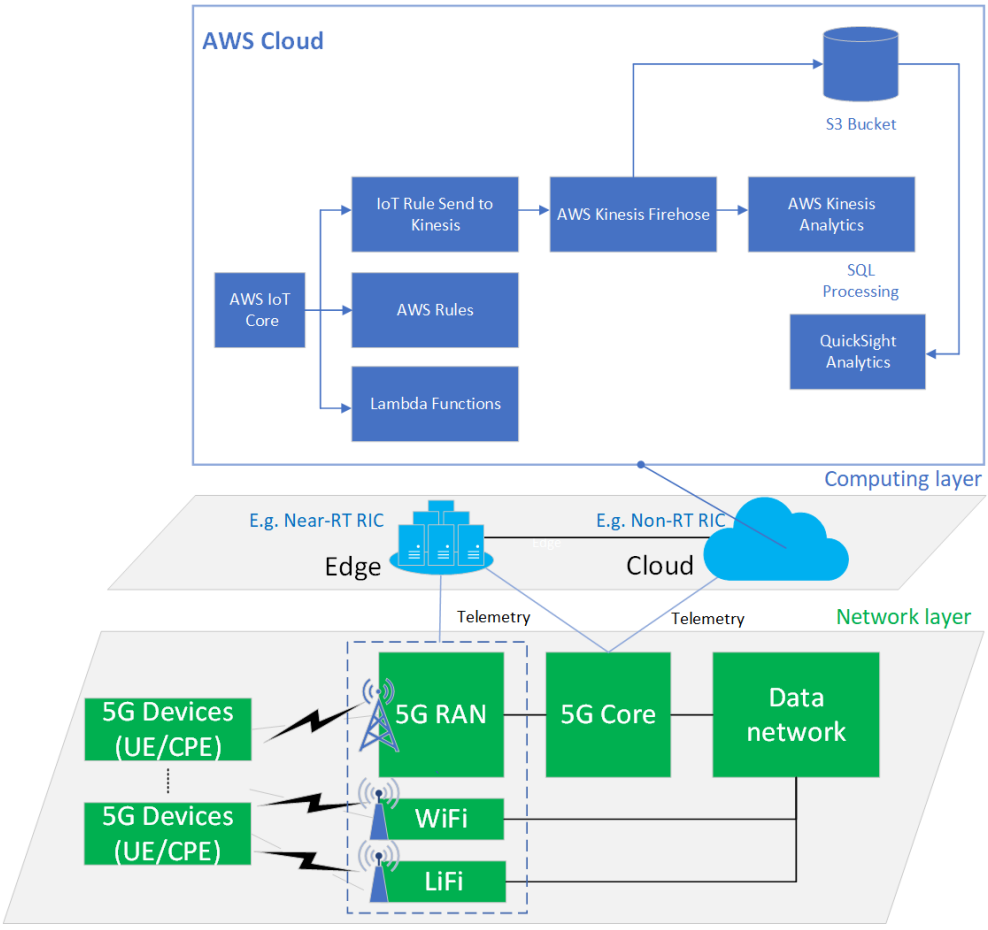


Figure 2.23 5G-CLARITY overall cloud-based solution for multi-WAT telemetry.

3 Integration between Private and Public Networks

A key feature of 5G-CLARITY system is the ability to facilitate private-public network integration, thereby allowing the realization of PNI-NPNs. A PNI-NPN is an E2E network composed of two networks: one *private*, consisting of VxFs that are provided by the private NOP; and one *public*, consisting of public VxFs that are provided by the public NOP. While private VxFs are typically executed on 5G-CLARITY infrastructure, public VxFs can be deployed using either 5G-CLARITY resources (i.e. on-premise resources) or PLMN resources (i.e. off-premise resources).

Based on the above rationale, it is clear that both the private NOP – responsible for providing private VNFs and managing 5G-CLARITY resources - and the public NOP – responsible for providing public VNFs and managing PLMN resources – play a key role in the provisioning of a PNI-NPN. To ensure a unified operation of this E2E network, it is thus required that the management systems of both NOPs interact with each other, exchanging trusted and verifiable messages between them. To facilitate the interaction between these systems, namely the 5G-CLARITY management and orchestration stratum (managed by the private NOP) and the 3GPP management system (managed by the public NOP), the network service aggregator role provides mechanisms leveraging on two key functionalities: **capability exposure** and **auditability**.

A first analysis on these functionalities were conducted in 5G-CLARITY D2.2 ([2], Section 9.3), leveraged from which this section delves into capability exposure functionality. In particular, the applicability of this functionality to relevant 5G-CLARITY service models, including WATaaS, NFVIaas and SlaaS (see 5G-CLARITY D2.2 [2], Annex B) are studied. The result of this study will be the identification and characterization of different management models, each providing a different interaction between 5G-CLARITY management system and 3GPP management system. Individual management models will be linked to real-world use cases to illustrate their usability. Further refinements of these management models and an initial design of auditability mechanisms will be specified in 5G-CLARITY D4.2.

3.1 5G-CLARITY slice management models

As introduced in 5G-CLARITY D2.2 the management system shall support a variety of management models. We discuss in this section how the 5G-CLARITY management framework will support each of these models, and when required how it will integrate with management systems of public networks.

The description of the management models relies on key definitions and concepts introduced in 5G-CLARITY D2.2. One is related to the characterization of the deployment scenarios, where a particular management model may be applicable or not. A deployment scenario can be identified by the number and type of actor roles, e.g. Network Operators (NOPs), service providers (SPs) or aggregators (AGGs). The actor roles involve different responsibilities regarding operations. In the context of 5G-CLARITY, they also determine the presence of both public and private administrative domains.

Another key feature of the deployment scenarios is related to the offered services. The 5G-CLARITY slice is defined as a concatenation of wireless, compute and transport services. The nature of the offered service (i.e. private or public) is another aspect that is representative from a business viewpoint. Accordingly, 5G-CLARITY infrastructure services can be intended for private use, such as an Industry 4.0 ultra-reliable low-latency service, or public use, such as mobile broadband for ultra-high definition video streaming.

Lastly, multiple service delivery models can be defined between different administrative domains, determining the interaction of actors from public and private domains in a given scenario. 3GPP has defined two main service delivery models for network slicing [81]: network Slices as NOP internals, where the existence of the network slices is transparent to the customer; and Network Slice as a Service (NSaaS), where

the network slice is offered as a communication service to the customer.

5G-CLARITY ecosystem opens doors to a great variety of service delivery models, whhich go well beyond 3GPP scope: WAT as a Service (WATaaS), NFV Infrastructure as a Service (NFVIaaS) and Slice as a Service (SlaaS). For example, the WATaaS model is an adaptation of the existing neutral hosting model, whereas the NFVIaaS model is used to make a virtual infrastructure available for VNF/VAF hosting. Finally, the SlaaS model is about making a 5G-CLARITY slice (i.e. on-premise infrastructure slice) available for a 5G-CLARITY tenant, so he can consume in their own administrative domain as desired.

To make services available for customer consumption, the above service delivery models leverages on the capability exposure. Capability exposure can be defined as the ability of a service provider to expose management capabilities to an authorized customer in a secure way. Management capability exposure in multi-tenant environments like the 5G-CLARITY brings some implications, particularly considering that different tenants could want to have different levels of management over their serving slices. This fact makes necessary to define different levels of exposure [2]. Looking at the literature, the 5G-VINNI project has proposed a level-based framework for capability exposure [82]. This framework lies on the definition of four capability exposure levels, each allowing the tenant to get a different set of operational capabilities from the service provider. As can be seen in Figure 3.1, the selection of one or another level allows a tenant to consume more or less capabilities from the management functions residing in the provider's administrative domain. For example, the first exposure level (i.e. the tenant retains control of network slice application layer configuration and management) only allows to the tenant to gain access to the E2E related management functions. However, if level 4 (i.e. the tenant retains control of virtualized resource control and management, scoping NFVI with optional Enhanced Platform Awareness (EPA) capabilities and infrastructural SDN control) is selected, then tenant can gain access up to the VIM. In other words, the tenant can consume operations and data offered not only by the VIM (level 4), but also by the NFVO (level 3), the management functions from individual network domains (level 2) and the E2E related management functions (level 1).

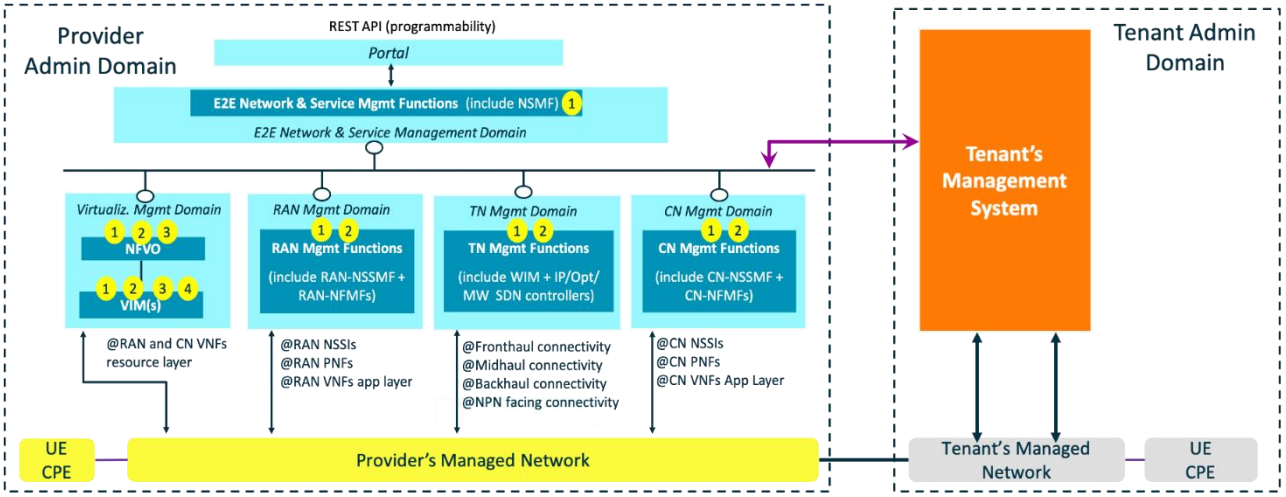


Figure 3.1: Mapping 5G-VINNI exposure levels into consumable capabilities in a baseline telco management and orchestration system.

In 5G-CLARITY, we firstly define the management model for the customer-facing and resource-facing services of a 5G-CLARITY facility infrastructure. Such models can be used for managing the exposed services according to the service delivery models (i.e. WATaaS, NFVIaaS and SlaaS). Then, we illustrate the use of the management models in some example deployment scenarios, together with the management entities that allow such exposure levels and interactions between actors from public and private domains. The role of the network service aggregator is also essential for this analysis since it determines how the federation is carried

out between public and private domains.

Table 3-1 describes the management models (MMs) defined in 5G-CLARITY for each service. This definition is made under two assumptions. One is that higher exposure levels mean more advanced management capabilities. The other statement is that the management capabilities exposed by a certain level also contains those capabilities offered by the lower levels. For the sake of readability, we use the notation *x.MMy*, where *x* represents the service (or the service delivery model) and *y* stands for the level of management capability exposure for that service. For each management level, the table describes which capabilities are available for the tenant and how much effort is expected for the service provider (SP) to manage the infrastructure. It is observed that the 5G-CLARITY transport service has not associated a service delivery model in the same way as other services (e.g. WATaaS, NFVIaaS). However, this service is offered indirectly as part of the SaaS model. When 5G-CLARITY offers an SlaaS, it is guaranteed a transport quota that enables the connection between the different services the slice is composed of (the transport service is responsible for the connection of the wireless and compute services).
Then, it is worth mentioning that while the NFVIaaS allows the tenant to extend its footprint, the SlaaS provides to the tenant a full slice that it can use to extend a certain functionality.

Table 3-1 Description of 5G-CLARITY Management Models.

Service	Related Service Delivery Model	Mgmt. Level	Description
5G-CLARITY Wireless Service	WATaaS	**WAT.MM1**	**Tenant**: manages FCAPS (streaming telemetry and data lake access) and the lifecycle of network slices (S-NSSAIs).
5G-CLARITY Compute Service	NFVIaaS	**NFVI.MM1**	**Tenant**: provides VNFDs/NSDs and software images for Application Functions (AFs) and manages FCAPS (streaming telemetry and data lake access)
		NFVI.MM2	**Tenant**: **NFVI.MM1** + management of lifecycle of VNFs/NSs (deploy, scale, heal, operate -start/stop/restart/…-, update and terminate, policies, etc.), onboarding of VNFDs/NSDs for NFs and AFs. The tenant connects to the private operator's NFVO to manage all the features aforementioned.
		NFVI.MM3	**Tenant**: **NFVI.MM2** + connection to SP's VIM through its own NFVO in order to have increased capability to manage VNF/NS lifecycle.
5G-CLARITY Network Slice[1]	SlaaS	**SL.MM1**	**Tenant**: supports the deployment of AFs, including **WAT.MM1** (S-NSSAI LCM) + **NFVI.MM1** (VNFDs/NSDs).
		SL.MM2	**Tenant**: manages the lifecycle of a 5G-CLARITY Slice, including WAT**.MM1** + **NFVI.MM2** (VNF/NS LCM).
		SL.MM3	**Tenant**: it includes WAT**.MM1** (S-NSSAI LCM) + **NFVI.MM3**[2] (connection to VIM).

The three SaaS described management levels in the table above include a guaranteed transport quota providing the connection of the different network services that conform the slice offered as a service, as it was aforementioned. Furthermore, all these management levels regarding the SlaaS delivery model require the interaction between the tenant's and private operator's slice managers.

[1] For simplicity, only MM levels for internal services to the 5G-CLARITY system are considered here.

The application of the management models varies depending on the deployment scenario and the presence or not of a public operator. In this way, some representative deployment scenarios have been selected for the analysis of the management models. We provide each scenario with a realistic example of a private venue (e.g. a factory, a museum, etc.) demanding a 5G-CLARITY facility infrastructure. We also analyse the actor roles (including the aggregator), the offered services, the service delivery models and the interactions between the different actors, especially public and private operators.

3.1.1 Scenario 1: NFVIaaS

This first scenario, described in Table 3-2 and Figure 3.2, represents the case in which the private operator (acting as the SP) offers NFVI as a service to a tenant, being the tenant in this case an MNO. With this service delivery model, the MNO can onboard some of its NFs on the private premises with the purpose of extending its service footprint and also to reduce latency. The MNO acquires a compute resource quota provided by the private operator with the aim of having an isolated execution environment in which the tenant can deploy its virtualized network functions (i.e. public network functions are deployed using on-premise resources).

Figure 3.2 illustrates the management and network views for this scenario. The green circle represents the service perceived by a tenant. The grey blocks represent the management entities, while the arrows indicate an action. This allows to show the different management models that are available in the scenario. In NFVIaaS there are different levels of management. Specifically, the MNO can choose between level 1 and 2. Whereas the management level 1 allows the tenant only to provide the VNFDs/NSDs and software images, with level 2 the MNO can manage the lifecycle of the network services hosted within the private premises (see Table 3-1). To do so, the tenant's MANO is able to connect to the MANO of the SP.

With the focus on facilitating the management tasks, some associations between the manager entity of the service provider and the tenant are required. For instance, there should be a connection between the private operator slice manager and the data semantic engine and data lake block to support the telemetry.
The museum acting as the private operator provisions the MNO with a set of compute resources which comprises a 5G-CLARITY slice, which remains inside the premises of the private venue (as it is observed in the bottom of Figure 3.2).
The WATaaS scenario would be similar to this one. In that case, the tenant could choose only one level of management, which is the one described in Table 3-1. This service delivery model enables the tenant to increase wireless capacity and coverage due to the wireless resources provisioned by the private operator.

Table 3-2 Description of a Network Scenario Offering NFVIaaS.

Aspect	Description
Actor Roles	• NOP+SP: Museum • Tenant: MNO
Services	Public: • MNO reducing latency
Service Delivery Models	• NFVIaaS (Museum → MNO)
Management Models	• Museum: NFVI.MMy(SP) • MNO: NFVI.MMy(Tenant) y ∈ {1,2}

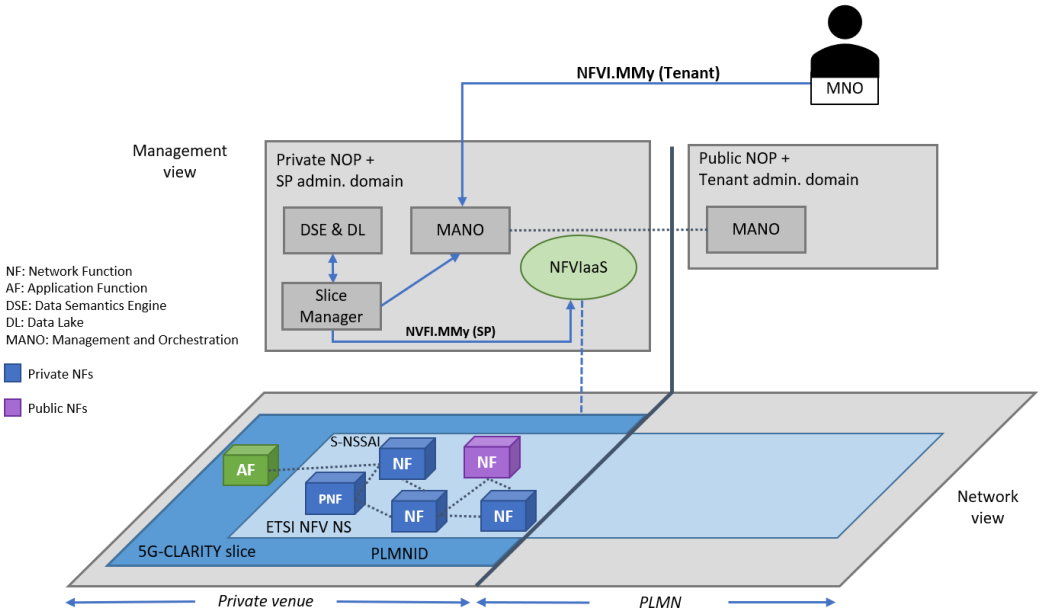


Figure 3.2 Management and network views of a scenario offering NFVIaaS.

3.1.2 Scenario 2: SNPN & SlaaS (e.g. a fully isolated factory)

The second scenario is a standalone non-public network, where the private owner offers a slice as a service. For example, a factory can offer a slice as a service to the factory Information Technology (IT) department that provides for instance virtual reality applications to that department workers (see the complete description in Table 3-3). Alternatively, the factory could subcontract the role of operator and provider to a micro-operator, or an MNO. In any case, the service delivery model is the 5G-CLARITY SlaaS. As observed in Figure 3.3, the slice is a 5G-CLARITY slice because it is fully deployed within the private venue. The green circles represent the slice as it is perceived by an actor. Accordingly, the 5G-CLARITY slice is offered as a service. The customer sees a slice that contains the whole functionality of a 5G system and is tailored to the specific necessities of its offered services. So that, the customer uses it to build a communication service. The figure also allows us to show the different management models that are available in the scenario. For example, levels 1, 2 or 3 of SaaS can be offered by the factory, which takes the roles of network operator and service provider. In the other side, the factory IT department, which uses the same level 1, 2 or 3 (represented by the letter y), operates as a tenant (that is the SlaaS service customer).

Some associations between the manager entity of the service customer and the provider are required. In particular, there should be a connection between the tenant slice manager with the slice manager of the 5G-CLARITY system. In addition, there should be a connection with the data semantic engine and data lake block to support the telemetry.

In the bottom of Figure 3.3, the network view represents the 5G-CLARITY slice, whose identification is given by a PLMNID-NID, corresponding to a standalone scenario. In addition, all the network elements are within the private venue.

Table 3-3 Description of a SNPN Scenario Offering SlaaS.

Aspect	Description
Actor Roles	• NOP+SP: factory • Tenant: factory IT department
Services	Private:

	• Factory IT department providing services and applications to workers
Service Delivery Models	• SaaS (factory factory IT department)
Management Models	• Factory: SL.MMy(SP) • Factory IT department (tenant): SL.MMy(Tenant) y ϵ {1,2,3}

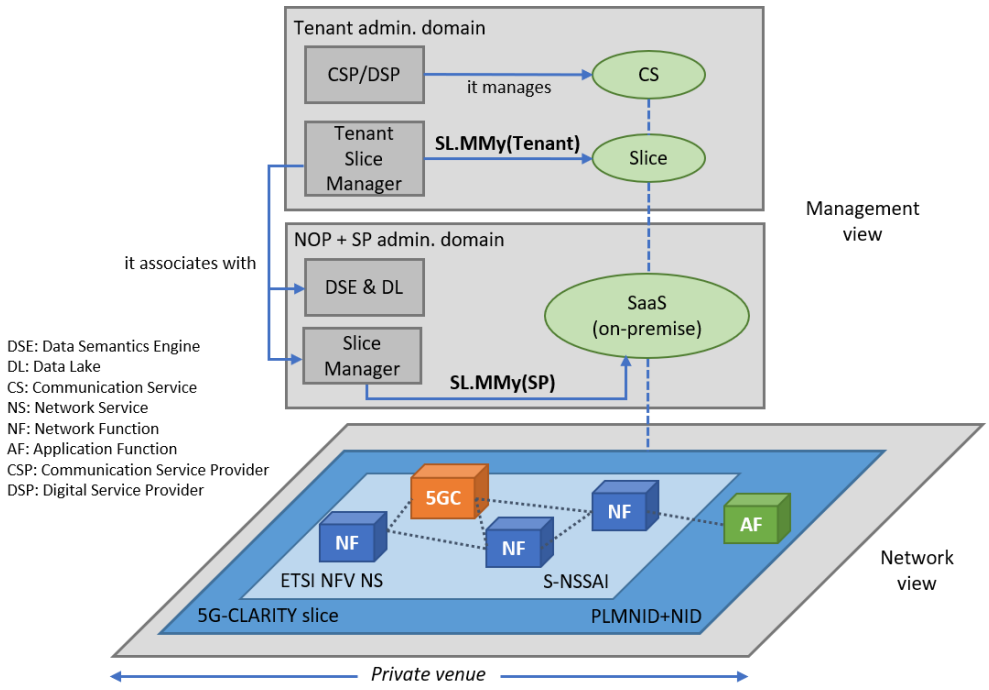


Figure 3.3 Management and network views of a SNPN scenario offering SlaaS.

3.1.3 Scenario 3: PNI-NPN & SlaaS (e.g. a stadium supported by an MNO)

In this scenario, represented in Table 3-4 and Figure 3.4, the actor setting up the 5G-CLARITY slice is the MNO and not the private operator. In particular, the private operator can be the owner of a stadium, the MNO, acting as a tenant, can also take the role of aggregator and another tenant can use the service offered by the MNO. An example of this second tenant could be a company that provides immersive sports applications to fans.

Looking at the green circles in the figure, the stadium offers a SlaaS using wireless, compute and transport services. As a provider, the private operator takes the level 3 of management for SlaaS. The MNO receives the 5G-CLARITY slice provided by the stadium and builds a slice including other network functions, for example, the 5G-Core. On the top of this slice is where the second tenant (e.g., the AR/VR company, as stated in Table 3-4) builds its services. The MNO uses the level 3 of SlaaS from the perspective of the service customer. In turn, it offers the new PNI-NPN slice made from the 5G-CLARITY slice to the second tenant using the level 1 of SlaaS, which means high effort for managing the slice as a provider. In the other side, the tenant utilizes the level 1 of SlaaS taking the role of customer, meaning that the level of exposure is low. Thus, there are two service delivery models in the scenario, but only a 5G-CLARITY slice, which takes part of the slice the MNO offers to another tenant. In this way, the MNO would require similar capabilities to the 5G-CLARITY Slice Manager in order to operate 5G-CLARITY slices.

Table 3-4 Description of a PNI-NPN Scenario Offering SlaaS.

Aspect	Description
Actor Roles	• NOP+SP: Stadium • NOP+Tenant+AGG+SP: MNO • Tenant: AR/VR company
Services	Private: • AR/VR company providing immersive sports applications to fans Public (optional, not shown): • MNO increasing wireless capacity to meet the demand of public subscribers
Service Delivery Models	• SaaS (Stadium → MNO) • NSaaS (MNO → AR/VR company)
Management Models	• Stadium: SL.MM3(SP) • MNO: SL.MM3(Tenant), SL.MM1(SP) • AR/VR company (tenant): SL.MM1(Tenant)

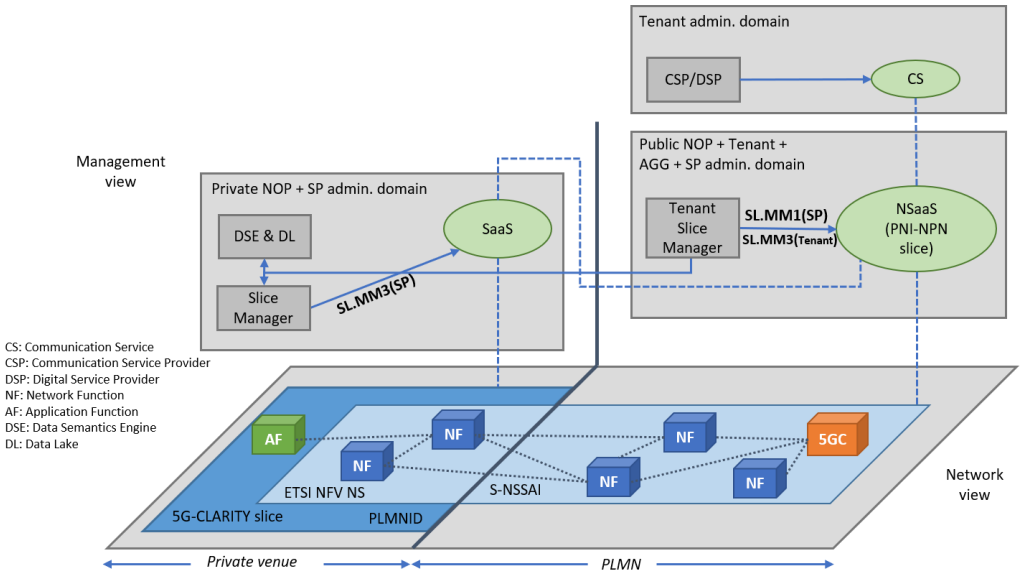


Figure 3.4 Management and network views of a PNI-NPN scenario offering SlaaS.

4 5G-CLARITY ML Algorithms

This section describes the machine learning (ML) use cases that are contributed by various partners of the 5G-CLARITY consortium, including the proposed ML algorithms.

4.1 Introduction

The 5G-CLARITY project has a strong focus on leveraging machine learning to support autonomous network management from different perspectives. Sections 4.2 to 4.10 describe nine ML use cases from AT3S handover to SLA violation and resource provisioning, using a range of ML algorithms that include Support Vector Machine and Deep Reinforcement Learning. Because of the sparsity of available suitable training data from real networks, some of these algorithms will initially be trained in a simulation-type environment and later deployed in the 5G-CLARITY AI engine as ML services that consume data from the 5G-CLARITY Data Management component and provide predictions for 5G-CLARITY management and network functions.

4.2 Predicting SLA violations/success rate

4.2.1 Problem statement

During the evolution of mobile networks, there has been an increasing demand not only on data traffic but also on various types of services such as reliability and latency for uRLLC. Each of these services has their own requirements that should be satisfied at the same time along with other types of services. As 3G and 4G networks are designed to accommodate increasing user data demand, satisfying QoS KPIs for different services is a challenging task for such dedicated networks. Therefore, 5G networks have been being designed in a way that efficiently operates multiple virtual sub-networks that sit on the same physical infrastructure. This is known as “network slicing” and has opened various opportunities for new services and use cases for MNOs as well as infrastructure owners. An example use case can be that an MNO (public operator) is a tenant and leases part of the network infrastructure for a specific time from the infrastructure owner (private operator). As this is a business model for both parties, there should be a set of specific business requirements that guarantees the capability of a network slice. This is termed as SLA in 5G terminology. As the network infrastructure has limited physical resources that are shared by several tenants, its resources should be carefully shared to satisfy SLAs. Therefore, network characteristics such as user demand, traffic types, spatial load distribution to access nodes and mobility patterns should be forecasted in order to predict possible SLA violations or success rate. A probability margin for possible SLA violations or success rate could be used by private/public operator to decide whether initiate a service/slice or not.

4.2.2 State of the art

Resource management in a heterogeneous network that provides diverse KPIs for different service types is a challenging task due to dynamicity of the network parameters such as channel variations, user mobility, load distribution and service variations. This is also the case of SLA monitoring for network slices. With the recent advancements on ML and its applications to mobile networks, applying ML methods have become a viable option to forecast dynamic network parameters.

A very first study that uses learning for network traffic prediction is proposed in [83]. The purpose in [83] is to investigate spatial and temporal dependencies of network traffic among base stations. For this purpose, a hybrid deep learning model that consists of three components namely local and global encoders and long short-term memory (LSTM) is used to perform spatiotemporal modelling and prediction of network data traffic for each cell. The idea of using LSTM which is a modified version of recurrent neural network (RNN) is to consider long-term temporal dependencies by using back propagation. A network slice broker solution

that forecasts network capacity and schedules network slice in a way to satisfy SLA is proposed in [84]. Different from [83] where the data traffic is predicted on a cell basis, the traffic forecasting process is tenant/slice-based in [84]. One of the time series approach named Holt-Winters is used to predict the traffic type of the network slices. Once the slice traffic is predicted, an admission control algorithm which is based on geometric knapsack problem is run within the proposed network slice broker to either grant the network slice request or not according to resource availability. If the slice is admitted by the slice broker, then a multi-class slice scheduler is used to provide SLA of the slice via online reinforcement learning (RL). The work in [84] is extended by including spatial distribution of tenants to the slice forecasting process in [85].

A deep neural network architecture named DeepCog that considers a cost-aware traffic forecast is presented in [86]. The objective in [86] is to maximize revenue of the network operators by reallocating long-term and short-term scheduling decisions that would help to reduce demand overprovisioning. Different from the noted traffic forecasting studies, DeepCog targets capacity prediction rather than mobile traffic forecast. The hyperparameters of the considered deep neural network are tuned via extensive simulations and tests. Another deep learning-based approach to provide SLA requirements is proposed in [87]. A soft gated recurrent unit (GRU) based traffic prediction is introduced to forecast the network slice traffic. GRU has the same objective as LSTM which is to consider long term dependencies. The difference of LSTM and GRU is the way both approaches operate. LSTM uses input, forget and output gates, whereas GRU uses reset and update gates, hence, simpler than LSTM [88] Once the slice-based traffic is predicted, SLA-constrained deep neural networks are used to estimate the required resources to not violate the SLA. A lower bound of the convergence probability of the SLA-constrained deep neural network is also provided.

In 5G-CLARITY, the proposed solution for predicting the network traffic/load for the overall network traffic volume and its spatial distribution to cells as well as obtaining the possible SLA violations/success rate probability/margin is based on echo state networks (ESNs). ESNs are special class of RNNs and are used to learn black-box models of non-linear systems such as the time series prediction tasks, supervised training of temporal pattern recognition, pattern generation, prediction, controller and more [89]. They have a non-trainable sparse recurrent part which is termed as reservoir and a simple linear readout. The input and reservoir weights are randomly generated and only the readout from the reservoir is trained [90]. The echo state property of ESNs comes from the reservoir in a way that the reservoir exhibits a fading of the input. In other words, the reservoir exhibits a short-term memory of the input that can be used to classify dynamic patterns.

4.2.3 5G-CLARITY initial design

Let's assume that one of the tenants of the private network operator wants to initiate a slice in the private network. Before initiating the slice request, the private network operator may want to predict a possible SLA violation/success rate for this service. As noted, the considered ML model for predicting the spatial distribution of the network traffic and obtaining the possible SLA violations is based on ESN which is a special class of RNN. It is important to note that not every randomly generated RNN has the echo state property of ESNs [89]. In order to satisfy the echo state property of the reservoir, the spectral radius of the reservoir weight matrix should be smaller than 1 [91].

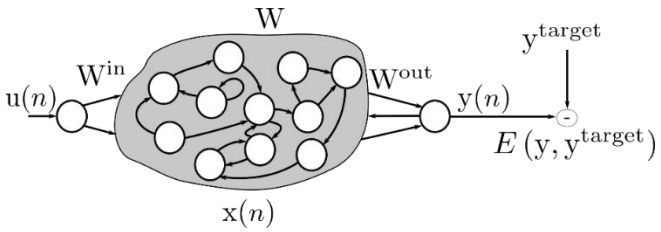


Figure 4.1: A high-level representation of an ESN architecture.

A high-level representation of the ESN architecture is depicted in Figure 4.1 [91]. The ESN architecture consists of an input signal $\mathbf{u}(n) \in \mathbb{R}^{N_u}$, a vector of reservoir activations $\mathbf{x}(n) \in \mathbb{R}^{N_x}$, an output vector of $\mathbf{y}(n) \in \mathbb{R}^{N_y}$ and a desired target output signal of $\mathbf{y}^{\text{target}}(n) \in \mathbb{R}^{N_y}$ where N_u, N_x, and N_y represent the number of elements in vectors $\mathbf{u}$, $\mathbf{x}$, and $\mathbf{y}/\mathbf{y}^{\text{target}}$, respectively; and $n = 1, \ldots, T$ is the discrete time with T number of data points. The $\mathbf{W}^{\text{in}} \in \mathbb{R}^{N_x \times (1+N_u)}$, $\mathbf{W} \in \mathbb{R}^{N_x \times N_x}$ and $\mathbf{W}^{\text{out}} \in \mathbb{R}^{N_y \times (1+N_u+N_x)}$ represent the input, recurrent and output weight matrices, respectively. The goal of the ESN is to minimize an error measure $E(\mathbf{y}, \mathbf{y}^{\text{target}})$ by learning a model that outputs $\mathbf{y}(n)$. The error measure $E(\mathbf{y}, \mathbf{y}^{\text{target}})$ can be obtained as a root mean-square error as:

$$E(\mathbf{y}, \mathbf{y}^{\text{target}}) = \frac{1}{N_y} \sum_{i=1}^{N_y} \sqrt{\frac{1}{T} \sum_{n=1}^{T} \left(y_i(n) - y_i^{\text{target}}(n) \right)^2}.$$

The ESN state update equation can be written as:

$$\mathbf{x}(n) = (1 - \alpha)\mathbf{x}(n-1) + \alpha \tilde{\mathbf{x}}(n),$$

where α is the leaking rate which determines the speed of the reservoir update dynamics; and $\tilde{\mathbf{x}}(n) \in \mathbb{R}^{N_x}$ is a reservoir activation function and equals to

$$\tilde{\mathbf{x}}(n) = \tanh\left(\mathbf{W}^{\text{in}}[1; \mathbf{u}(n)] + \mathbf{W}\mathbf{x}(n-1) \right),$$

where tanh is the hyperbolic tangent function; and $[a; b]$ represents vertical concatenation of the vectors/matrices *a* and *b*. Once the ESN state is updated, the ESN output vector, readout vector, can be written as:

$$\mathbf{y}(n) = \mathbf{W}^{\text{out}}\mathbf{X},$$

where $\mathbf{X} = [1; \mathbf{u}(n); \mathbf{x}(n)]$ is a concatenation matrix of $\mathbf{u}(n)$ and $\mathbf{x}(n)$. The optimal weights of $\mathbf{W}^{\text{out}}$ can be obtained as:

$$\mathbf{W}^{\text{out}} = \mathbf{y}^{\text{target}}\mathbf{X}^{\boldsymbol{T}}\left(\mathbf{X}\mathbf{X}^{\boldsymbol{T}} + \beta\mathbf{I}\right)^{-\mathbf{1}},$$

where β is a regularization coefficient that is being used to mitigate overfitting or feedback instability [91]; and $\mathbf{I}$ is the identity matrix.

In the case of using a feedback from the readout to reservoir, the reservoir activation function can be re-written as:

$$\tilde{\mathbf{x}}^{\text{feedback}}(n) = \tanh\left(\mathbf{W}^{\text{in}}[1; \mathbf{u}(n)] + \mathbf{W}\mathbf{x}(n-1) + \mathbf{W}^{\text{feedback}}\mathbf{y}(n-1) \right),$$

where $\mathbf{W}^{\text{feedback}} \in \mathbb{R}^{N_x \times N_y}$ is the feedback weight matrix.

In order to build the ESN model for spatiotemporal network traffic distribution and SLA violation/success rate prediction, a reservoir RNN with N_x number of neurons should be generated along with random input ($\mathbf{W}^{\text{in}}$) and reservoir ($\mathbf{W}$) weight matrices. The training input $\mathbf{u}(n)$ which in general is normalized to have bounded values and includes traffic volume, spatial volume distribution, traffic class and its performance and user mobility pattern information should be fed into the RNN to collect reservoir activation states $\mathbf{x}(n)$. Then, the weights of the output ($\mathbf{W}^{\text{out}}$) should be computed in order to minimize the error measure between the output data and the desired data as presented by $E(\mathbf{y}, \mathbf{y}^{\text{target}})$. The output of the ESN should represent a probability distribution of an SST which can be eMBB, URLLC etc. and its corresponding performance based on the predicted network load. Once this output is evaluated, the tenant-specific SLAs are used to evaluate a probability margin that represents a violation or success rate of the considered SST. This prediction then can be used by the private operator or its tenant to initiate this service or not. As this is a non-RT process, it can be initiated any time by the operator.

As in any other ML model, the ESN has a set of parameters that need to be defined in order to improve the performance of the model. These parameters can be listed as, (i) the size of the reservoir N_x; (ii) the sparsity of the reservoir; (iii) the spectral radius of the reservoir weight matrix $\rho(\mathbf{W})$; (iv) the scaling of the input weight matrix $\mathbf{W}^{\text{in}}$; (v) the leaking rate α; and (vi) the regularization coefficient β. The effect and interaction among these parameters are well defined in [91] and will be identified for network traffic forecast and SLA violation/success rate in the next deliverables of WP4 in 5G-CLARITY.

The proposed approach will be evaluated as follows. Firstly, a data set will be generated via system-level simulations in MATLAB and Mininet/NS3. This data set will include time stamped data of (i) the aggregated throughput per UE that is reported by its connected gNB, Wi-Fi AP and/or LiFi AP; (ii) the total load offered by each cell; (iii) the service type details including the success rate/performance; and (iv) the connected cell ID of the users. Once the data set is collected, it will be partitioned and used for training and validation purposes such as 80% of the data will be used for training and the remaining 20% will be used to validate the performance of the ESN.

4.3 RT RIC: AT3S traffic routing/handover

4.3.1 Problem statement

A path selection is required after a Multi-access (MA) PDU Session has been established and two user plane paths are available. That is, a UE should choose over which access each packet will be transmitted. Moreover, the UPF should choose which access to be used for each packet. A network function that controls both the UE and the UPF in choosing the paths in the PCF in the form of PCC rules. Then, the SMF derives AT3S rules for the UE and N4 rules for the UPF. Both rules are for controlling the traffic steering, switching, and splitting in the UE and the UPF, respectively. Figure 4.2 shows the relationship between the UE, the UPF, the PCF, and the SMF.

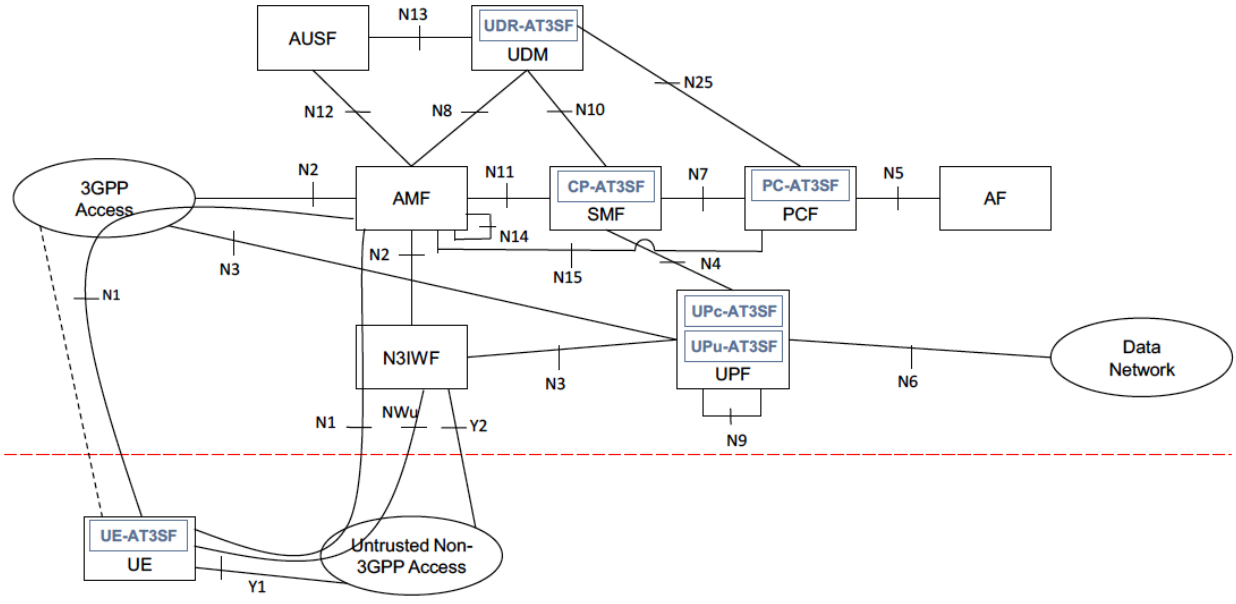


Figure 4.2 High level AT3S architecture [92].

The AT3S rules include:

- a precedence value, which denotes the priority of the AT3S rule of the current UE with respect to

those of other UEs,

- a traffic descriptor, which details a service data flow (SDF),
- a steering mode, which identifies an approach (i.e., 'Active-Standby', 'Smallest Delay', 'Load-Balancing', and 'Priority-Based') of how a SDF will be steered,
- a steering function, which controls whether the higher layer steering function or the lower layer steering function are used, and,
- an indicator to support the 3GPP RAT differentiation.

The N4 rules also include many descriptors, e.g., precedence value, forwarding action value, or multi-access rule, see Table 6.14.1 in [92].

The AT3S and N4 rules such as steering mode, priority level or load balance percentage need to be updated during the ongoing MA PDU session, and they depend on the condition of the networks. This is where an ML algorithm comes into help. It can automatically update the rules or parameters within the existent rules in an intelligent manner.

4.3.2 State of the art

There is still not a work that directly focuses on applying ML algorithms to a complete AT3S functionality as described in the previous subsection. However, many works have discussed it partially. For example, the authors in [93] try to mitigate the greedy nature of MPTCP by means of predicting the traffic by using SVM. In the context of determining the optimal policy generated by PCF that is kept being updated about the network condition, the RL is a more appropriate ML technique to be used. An example of such work is discussed in [94], where a deep RL is used with recurrent neural networks. The application of RL is not only limited in the PCF, but also a RL-based scheduler is feasible, such as in [95].

In [92], there are two different steering functions, i.e., the higher layer steering function (referred to as MPTCP) and the lower layer steering function (referred to as AT3S function). In this context, Figure 4.3 depicts a structure in a UE that supports both MPTCP and AT3S functions. Unlike the MPTCP function, few works focus on the AT3S function. The main difference between both functions is that the AT3S function only supports the packet switching. Moreover, the AT3S function can handle all types of traffic, e.g., IP (TCP and UDP) or Ethernet traffic [92]. Other steering methods are still under discussion in IETF, such as the trailer-based encapsulation protocol [96]. In the same layer as that of the AT3S function, another potential protocol is the Fast Session Transfer (FST) protocol of the IEEE 802.11ad or the IEEE 802.11ay. The FST protocol is relevant here as the 802.11ad occupies the mmWave band as in the 5GNR mmWave.

Based on the previous paragraph, there are many low hanging fruits on the AT3S functions. Within a UE, the AT3S function is responsible of choosing either 3GPP or non-3GPP accesses. Once the UE receives the AT3S rules from the SMF telling the UE to use the AT3S function, the UE has the full authority to select either of them. Many studies fall into similar problem, i.e., how to optimally select WATs. For example, the authors in [97] study such problem from the point of view of multi-armed bandit problems, which are a classic problem for the RL. A context-aware, optimal network selection method is proposed in [98]. Based on [98], the WAT selection can consider the asymmetry of uplink and downlink, the traffic-type-location-type of information, and the load changes. The WAT selection problem can also be translated to an uncooperative game problem [99]. This approach can be potentially improved by means of multi-agent RL based on the seminal work in [100].

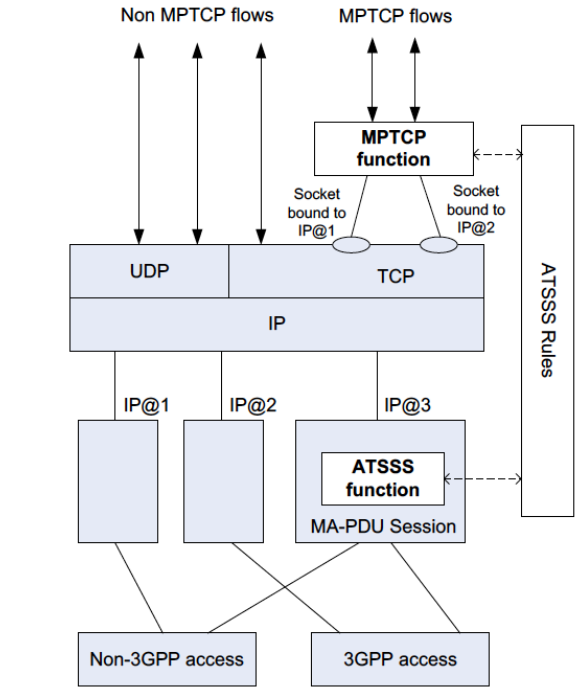


Figure 4.3 A structure of a UE that supports both MPTCP and AT3S functions [92] .

4.3.3 5G-CLARITY initial design

In this section, we focus on the congestion control in MPTCP for AT3S. To be more specific, we are interested in the study of the improvement of single deep reinforcement learning (DRL) agent to determine an optimal N4 rule for an MPTCP-enabled UPF. Then, the UPF feedbacks rewards and states to the DRL agent as illustrated in Figure 4.4. The proposed model compared to [94] is that we integrate a model-based system to a model-free DRL system. A model-based system can use, for example, a Wi-Fi, LiFi, 5GNR channel models. By using this hybrid approach, a system performance can be improved due to the combination of a powerful DRL algorithm and an insightful model-based system [101].

Suppose $\mathbf{p}_{t+1}$ is a position vector or a waypoint of a user at t+1 as shown in Figure 4.5, which can be predicted by random mobility model such as the modified Levy-walk model, then we can estimate the path loss of a Wi-Fi, a LiFi, or a 5GNR link. Provided a transmit power, the RSSI can be estimated, which then can be used to estimate throughput. By using the new estimated position vector, a blockage can be predicted, and the DRL agent can anticipate it. Generally, the DRL agent can benefit from an output of a function of $\mathbf{p}_{t+1}$, which is denoted by $f(\mathbf{p}_{t+1})$ as shown in Figure 4.5. Therefore, based on these predicted metrics, the DRL agent can devise a N4 rule for the UPF such as modifying the percentage of the traffic that goes through a specific technology in a way that generates a better reward compared to the model-free-only DRL system.

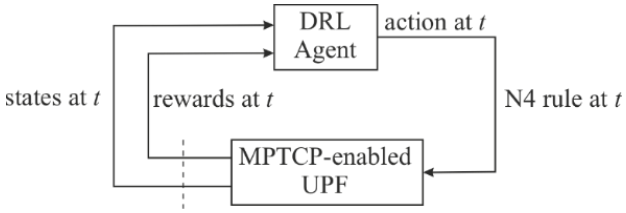


Figure 4.4 A high level illustration of the state-reward-action flow between an DRL agent and an MPTCP-enabled UPF.

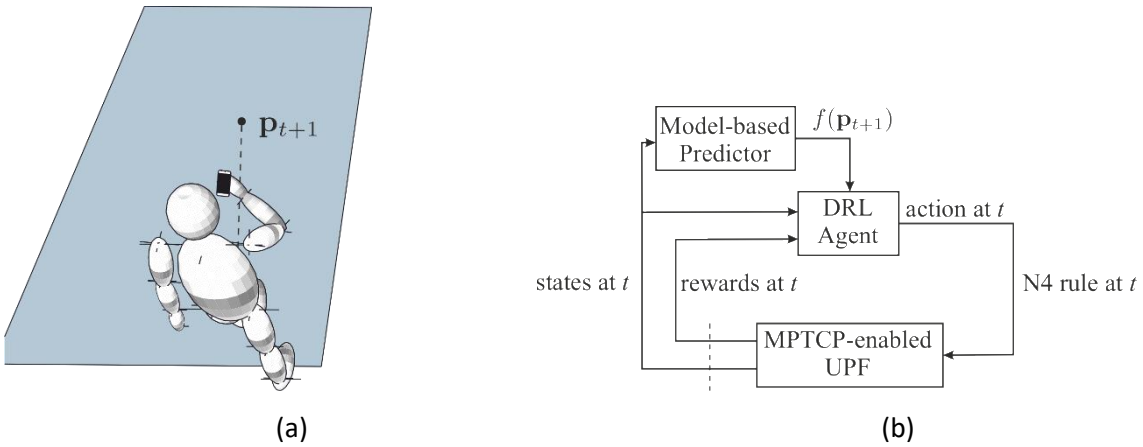


Figure 4.5 (a) An illustration of a new position vector that is generated by a model-based system, and (b) a hybrid of model-free and model-based DRL system.

As for our problem formulation, a Markov decision process (MDP) is defined with the addition of metrics that are function of the predicted position vector $\mathbf{p}_{t+1}$, such as predicted RSSI, blockage event, predicted throughput. In addition, the reward function for the MDP can also consider a factor that indicates the difference between the actual position at *t*+1 and the predicted position at *t*+1. This factor does not need to be calculated directly by using the position vector, which might indicate the need of a positioning system. However, it can also be inferred by the difference between, for example, the predicted and the actual RSSIs. With this additional reward function, the model that is used to predict the new position vector can also be improved.

In summary, the states that are used by the DRL agent can represent the goodput, RTT, congestion window size, the number of active MPTCP subflows, and the predicted metric from the model-based predictor. The reward function can be the average goodput from all MPTCP subflows. The actions can be the parameter inside the N4 rule, which can further define the new number of MPTCP subflows, congestion windows, or the splitting ratio for the packets.

During the running time, the DRL agent processes the rewards and the states every time sampling T_s. This time sampling cannot be too small since the DRL agent will learn static information. However, it cannot be too large as well since the DRL agent might lose important information. The initial plan is to define it as a fraction of coherence time. As the time of writing, the coherence time of RF channels is thoroughly investigated, but only a few studies are available for LiFi channels. Therefore, we plan to investigate the coherence time of LiFi channels, and then apply this information to our proposed DRL system.

In order to evaluate the proposed system, we use an emulator. Mininet-Wi-Fi [102] can be used to emulate the TCP/IP stack (including the MPTCP) and a Wi-Fi channel model. The mininet-Wi-Fi can then be extended with the LiFi functionality. First, a physical layer abstraction of LiFi channels should be defined. In defining the physical layer abstraction for LiFi channels, we use our simulator, named owcsimpy [103], which can define a realistic geometry model for indoor LiFi channels and generate CIRs. These Wi-Fi and LiFi models can then be integrated with an existing 5G emulator, e.g., a 5G module in ns-3 [104]. Then, the integration with the 5G module of ns-3 can be done based on [105]. By using Wi-Fi, LiFi, and 5GNR channel models and the emulators, we can emulate the proposed model-free and model-based DRL systems for controlling congestion for the MPTCP-enabled UPF.

4.4 RAN slicing in multi-tenant networks

4.4.1 Problem statement

The considered scenario assumes a private venue network owner of a NG-RAN infrastructure composed of *N* cells with diverse deployment characteristics (i.e., access technology, cell radius, transmission power, frequency of operation). Each cell *n* has a total of $N_T(n)$ physical resources, which provide a total cell capacity $C_T(n)$ (b/s). These physical resources depend on the specific access technology used by the cell, e.g. Physical Resource Blocks (PRBs) for the case of 5G New Radio (NR), airtime in case of Wi-Fi, etc. The network is shared among *K* different tenants, each of them provided with a RAN Slice Instance (RSI). The different tenants can be for example different MNOs that provide service to their own users through the private network following a neutral host model (see 5G-CLARITY D2.2 Section 3.3).

The considered problem consists in determining how the available capacity is distributed among the different RAN slices in the different cells. Given the heterogeneity and dynamics of the traffic (i.e. the traffic of the different tenants will vary in time and space and may exhibit complementarities in the different cells), a smart capacity sharing strategy will be proposed, which will dynamically determine the resource quota (i.e. the proportion of physical resources) allocated to each RAN slice in each cell and configure the network accordingly.

The objectives of the capacity sharing approach are two-fold. On the one hand, it needs to achieve an efficient utilisation of the available resources, avoiding both over and under provisioning. On the other hand, it needs to fulfil the SLA established between the private venue network owner and each tenant. This is assumed to be defined in terms of: (i) the Scenario Aggregated Guaranteed Bit Rate *SAGBR*(*k*) that measures the aggregate bit rate to be provided to tenant *k*, if requested, across all the cells in the network; and (ii) The Maximum Cell Bit Rate *MCBR*(*k, n*) that specifies the maximum bit rate that can be provided to tenant *k* in cell *n*. This limit is defined in order to avoid that all the capacity of a cell is assigned to a single tenant under highly extreme heterogeneous spatial load distributions with tenants demanding excessive capacity in certain cells. In this way a fair network resource sharing among tenants can be achieved.

4.4.2 State of the art

The capacity sharing problem for RAN slicing has been addressed by different works in the literature following different approaches. In some works, the problem has been dealt by means of mathematical optimisation, like in [106], which used the Karush Kuhn Tucker conditions, in [107], which used integer programming, or in [108], which used a biconvex problem for developing a joint share solution of radio resources, caching and backhaul. Similarly, in [109] the capacity sharing is achieved by regulating the number of admitted QoS flows of each tenant through an admission control optimized by means of a Semi-Markov Decision Process.

Other works approached the capacity sharing by means of heuristic algorithms. These include the exponential smoothing model in [110], the admission control approach based on profit-related metrics in [111], a market oriented model in [112], a winner bid problem in [113], a fisher market game in [114] or an iterative algorithm in [115].

In order to deal with the inherent uncertainty in wireless environment and the large-scale of 5G mobile networks [116], some other works have approached the capacity sharing problem in RAN slicing scenarios by using DRL algorithms [117]. In this regard, in [118] and [119], the capacity reserved to each slice is provided by means of Deep Q-Network (DQN) and Deterministic Policy Gradients combined with K-Nearest Neighbours, respectively. Additionally, [120] and [121] provide the cell capacity share solution by firstly computing the aggregated capacity reserved to each slice at network level by using DQN and, then, applying a heuristic algorithm to obtain the cell capacity for each tenant.

The proposed solution for the capacity sharing problem in 5G-CLARITY is based on a Multi-Agent Reinforcement Learning (MARL) approach, where a collaborative MARL algorithm learns the capacity to be provided to each tenant by associating each agent in the MARL to a different tenant. In this way, tenants can be easily added/removed in the scenario just by adding or removing the corresponding agent. To the best of our knowledge, none of the previous works has focused on MARL to obtain the capacity sharing solution in RAN slicing scenarios considering a multi-cell perspective. In this respect, the recent work [122] also considers a multi agent approach based on Ape-X technique associating different agents (so-called actors) to different slices. However, the capacity assignment is done independently for each cell, while in the proposed approach the assignment takes into account jointly all the cells in the scenario in order to balance the spatial and temporal traffic heterogeneities among different tenants. Additionally, the proposed model addresses the capacity sharing function when considering the SLA defined as an aggregate capacity across the whole scenario in order to capture the total amount of capacity to be provided to a tenant. Instead, other approaches such as [118]-[122] just consider the SLA specified in terms of the QoS parameters defined at user/QoS flow level, but without enforcing any aggregate capacity per tenant.

4.4.3 5G-CLARITY initial design

The proposed RAN slicing function dynamically tunes the capacity share for each tenant and cell in order to adapt to the spatial and temporal traffic variations among different cells, minimise SLA breaches in the system and optimise the resource utilisation of the different cells in the system. The capacity share of tenant k at time step t is defined as $\boldsymbol{\alpha_t(k)}=[\alpha_t(k,1),\ldots, \alpha_t(k,n), \ldots, \alpha_t(k,N)]$, where each component $\alpha_t(k,n)$ is the resource quota assigned to tenant k in cell n given by the proportion of the total physical resources $N_T(n)$ in the cell provided to the tenant during time step t and ranges $0 \leq \alpha_t(k,n) \leq MCBR(k,n)/C_T(n)$. Note that the capacity share solution in a cell cannot exceed the total capacity of the cell, so that $\sum_{k=1}^{K} \alpha_t(k,n) \leq 1$. To adapt to the varying traffic demands, the capacity share $\boldsymbol{\alpha_t(k)}$ is updated periodically in time steps of duration Δt that will be in the order of minutes (the appropriate setting of Δt will be studied during the performance analysis stage).

In order to deal with the complexity of the computation of $\boldsymbol{\alpha_t(k)}$ in multi-cell scenarios, the solution is designed as a MARL where each RL agent is associated to a tenant k that learns the policy $\pi(k)$ to tune $\boldsymbol{\alpha_t(k)}$ dynamically by interacting with the environment. Given that the learning needs to be performed continuously from the network environment and large state and action spaces are expected, each agent k derives its policy $\pi(k)$ according to a DQN based algorithm as the RL method.

Figure 4.6 illustrates the proposed solution scheme. The RAN slicing management function at the AI engine is composed by the DQN agents of the different tenants. At each time step t, each agent obtains the state $\boldsymbol{s_t(k)}$ from the environment and, based on the policy $\pi(k)$, triggers an action $\boldsymbol{a_t(k)}$ to tune $\boldsymbol{\alpha_t(k)}$. Moreover, a reward signal $r_t(k)$ that reflects the obtained performance after the last action $\boldsymbol{\alpha_{t-1}(k)}$ is provided to the k-th agent. Since the actions of each tenant are triggered independently, a capacity sharing solution validation function is necessary to ensure that the selected actions in each cell n do not exceed the total cell capacity.

Figure 4.6 also depicts the inputs that are used by the proposed solution. On the one hand, the SLA terms, i.e. $SAGBR(k)$ and $MCBR(k,n)$ need to be provided by the private network operator when the RAN slice is created. On the other hand, the telemetry collector needs to gather the following inputs on a Δt time step basis: (i) fraction of physical resources used for data traffic by tenant k in cell n in time t, denoted as $\rho_t(k,n)$, (ii) offered load by tenant k in cell n in time t, denoted as $O_t(k,n)$ and understood as the capacity required by the tenant to transmit its data, and (iii) total throughput of tenant k across all the cells, denoted as $SThr_t(k)$. These measurements at time t correspond to average values measured over the time window $(t-\Delta t,t)$. These inputs are used by each agent to compute the state and the reward, as explained in detail in the following.

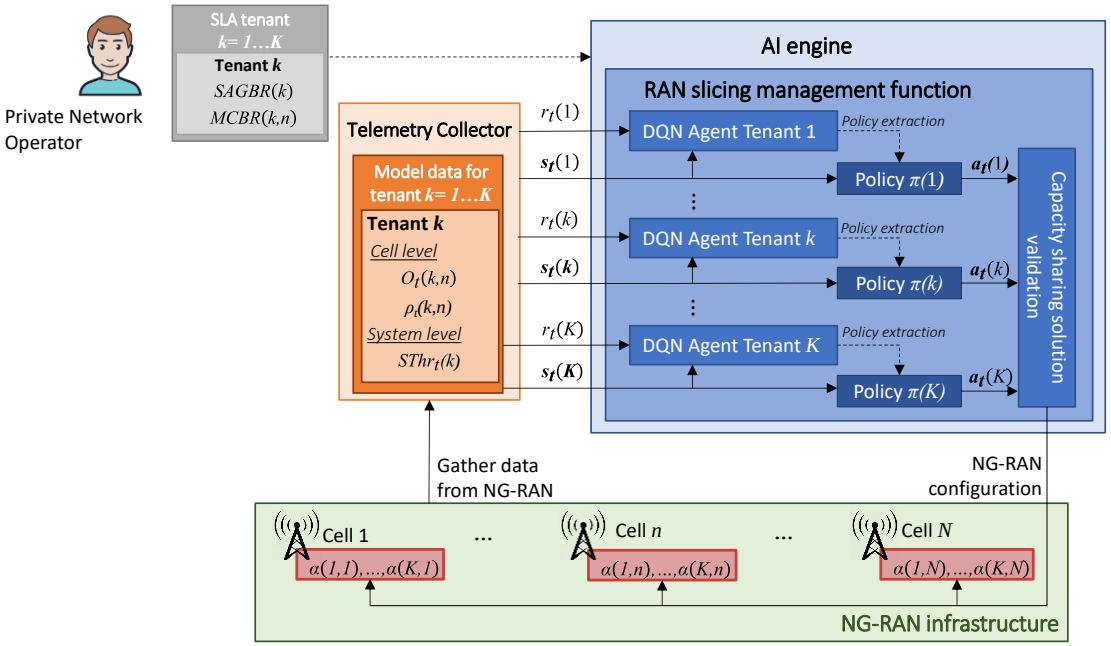


Figure 4.6: MARL scheme for RAN slicing.

a) State

The state of tenant k at time t is denoted as $\boldsymbol{s_t(k)}$ = [s_t $(k,1)$,…, $s_t(k,n)$, …, $s_t(k,N)$], where element $s_t(k,n)$ corresponds to cell n and is defined by the triple $<\rho_t(k,n), \alpha_t(k,n), \alpha_{ava,t}(n)>$, where $\alpha_{ava,t}(n)$ is the available capacity share in the cell not assigned to any tenant, given by $\alpha_{ava,t}(n)=1-\sum_{k=1}^{K}\alpha_t(k,n)$.

b) Action

The k-th tenant's agent triggers a joint action $\boldsymbol{a_t(k)}$ = [$a_t(k,1)$,…, $a_t(k,n)$, …, $a_t(k,N)$] for all the cells composed of the cell-specific actions $a_t(k,n)$, defined as the increase in capacity share $\alpha_t(k,n)$ of tenant k to be applied in the following time step in cell n. This increase is defined in discrete steps of size Δ, so that the action can take three different values $a_t(k,n)\epsilon\{\Delta,0,-\Delta\}$. Then, the capacity share to be applied becomes:

$$\alpha_t(k,n)= \alpha_{t-1}(k,n)+a_t(k,n) \quad (1)$$

Note that the action space for $\boldsymbol{a_t(k)}$ corresponds to all the possible combination vectors of the three possible actions for each of the cells. Then, the number of possible actions for each tenant is 3^N.

Initially, all components of $\boldsymbol{\alpha_{t=0}(k)}$ are initialized to:

$$\alpha_{t=0}(k,n)=SAGBR(k)/\left(\sum_{k=1}^{K} SAGBR(k)\right) \quad (2)$$

Then, the actions $\boldsymbol{a_t(k)}$ triggered by each of the tenants are considered and $\boldsymbol{\alpha_t(k)}$ is updated by applying (1) for each cell. In a specific case when the chosen action $a_t(k,n)$ forces $\alpha_t(k,n)$ to be out of its bounds (i.e. $\alpha_t(k,n)<0$ or $\alpha_t(k,n)>MCBR(k,n)/C_T(n)$), the last $\alpha_{t-1}(k,n)$ value is maintained instead.

c) Reward

The reward $r_t(k)$ assesses how good was the last action $\boldsymbol{a_{t-1}(k)}$ performed for the previous state $\boldsymbol{s_{t-1}(k)}$. The reward accounts for different components reflecting both the SLA satisfaction and the capacity utilisation.

The first component is the SLA satisfaction factor of tenant k, $\gamma_{SLA}(k)$, which is the ratio between the provided and the requested capacity, defined as:

$$\gamma_{SLA}(k)=min\left(\frac{SThr_t(k)}{min\left(\sum_{n=1}^{N} min(O_t(k,n),MCBR(k,n)),SAGBR(k)\right)},1\right). \quad (3)$$

The second component of the reward is the capacity utilisation factor, $\gamma_{ut}(k)$, which aims at minimising the overprovisioning and under provisioning of resources and is defined as:

$$\gamma_{ut}(k)=\frac{SThr_t(k)}{\sum_{n=1}^{N} C_T(n)\cdot\alpha_t(k,n)}. \quad (4)$$

Then, the reward $r_t(k)$ considers the weighted product of the SLA satisfaction factor of tenant k, the summation of the SLA satisfaction factors of the other tenants and the capacity utilisation factor of tenant k, that is:

$$r_t(k)=\cdot\gamma_{SLA}(k)^{\varphi_1}\cdot\left(\frac{1}{K-1}\sum_{\substack{k'=1\\k'\neq k}}^{K}\gamma_{SLA}(k')\right)^{\varphi_2}\cdot\gamma_{ut}(k)^{\varphi_3}, \quad (5)$$

where φ_1, φ_2 and φ_3 are the weights of each component.

d) DQN agent and training process

The DQN agent of tenant k centrally learns the policy $\pi(k)$ that dynamically configures $\boldsymbol{\alpha_t(k)}$. For this purpose, the original DQN algorithm in [123] has been particularised to the current state, action and reward definitions. The DQN aims at finding the optimal policy $\pi^*(k)$ that maximises the discounted cumulative future reward (i.e., $\sum_{j=0}^{\infty}\gamma^k r_{t+j+1}(k)$, where γ is the discount rate ranging $0\leq\gamma\leq1$), by obtaining the optimal action-value function $Q_k^*(\boldsymbol{s(k), a(k)})$, which is the maximum expected discounted cumulative reward starting at time step t from $\boldsymbol{s(k)}$, taking the action $\boldsymbol{a(k)}$ and following the policy $\pi(k)$:

$$Q_k^*(\boldsymbol{s(k),a(k)}) = E\left[r_{t+1}(k)+\gamma \max_{a'(k)} Q_k^*(\boldsymbol{s_{t+1}(k),a'(k)})\middle|\boldsymbol{s_t(k)=s(k), a_t(k)=a(k)}\right] \quad (6)$$

In DQN, $\boldsymbol{Q_k^*(s(k), a(k))}$ is approximated by a deep neural network (DNN) with weights θ, denoted as $Q_k(\boldsymbol{s(k),a(k)}, \theta)$. In order to update $Q_k(\boldsymbol{s(k),a(k)}, \theta)$, the DQN agent of the k-th tenant is composed of the following elements:

- **Evaluation DNN ($Q_k(\boldsymbol{s(k),a(k)}, \theta)$)**: corresponds to the main approximation function of the expected reward function $Q_k(\boldsymbol{s(k),a(k)})$.
- **Target DNN ($Q_k(\boldsymbol{s(k),a(k)}, \bar{\theta})$):** a separated DNN with weights $\bar{\theta}$ is used to obtain the Time Difference (TD)-Target, computed as $r_t(k)+\gamma \max_{\boldsymbol{a'(k)}} Q_k(\boldsymbol{s_t(k),a'(k)},\bar{\theta})$. This DNN is updated every M steps with the weights of the Evaluation DNN, i.e. $\bar{\theta}=\theta$. The architecture of $Q_k(\boldsymbol{s(k),a(k)},\bar{\theta})$ is the same as the one of $Q_k(\boldsymbol{s(k),a(k)},\theta)$.
- **Experience Dataset** $D_l(k)$**:** for each time step t, the experience tuple $e_t=<\boldsymbol{s_{t-1}(k)}, \boldsymbol{a_{t-1}(k)}, r_t(k), \boldsymbol{s_t(k)}>$ is stored into a dataset $D_l(k)$ of length l. Then, a mini-batch of experiences $U(D_l(k))$ of length J is randomly selected from $D_l(k)$ to update $Q_k(\boldsymbol{s(k),a(k)},\theta)$.

At initialisation, the weights of both the evaluation and target DNNs are randomly selected. Then, they are updated as a result of the training process of each DQN agent, which is divided in two parts: the data collection and the update of weights θ.

The data collection consists in gathering experiences from the network environment and storing them in the experience dataset $D_l(k)$. For each time step t, the DQN agent observes the state of the environment $\boldsymbol{s_t(k)}$ and, accordingly, triggers an action $\boldsymbol{a_t(k)}$ to the environment based on an ε-Greedy policy $\pi(k)$.This policy chooses actions according to $\boldsymbol{a(k)}=\underset{\boldsymbol{a'(k)}}{argmax}\, Q_k(\boldsymbol{s(k),a'(k)},\theta)$ with probability 1- ε and a random action with probability ε. Then, the reward $r_t(k)$ associated to the last performed action $\boldsymbol{a_{t-1}(k)}$ for the last state $\boldsymbol{s_{t-1}(k)}$ is obtained. The experience tuple $e_t=<\boldsymbol{s_{t-1}(k)}, \boldsymbol{a_{t-1}(k)}, r_t(k), \boldsymbol{s_t(k)}>$ is stored in the dataset $D_l(k)$. When the dataset $D_l(k)$ is full (i.e. l experiences are stored), old experiences are removed from the dataset to save new ones.

In parallel, the process of updating the weights q of $Q_k(\boldsymbol{s(k)},\boldsymbol{a(k)},\theta)$ is performed iteratively. Firstly, a minibatch of experiences $U(D_l(k))$, which has a length of J experiences, is randomly selected from the dataset $D_l(k)$. Then, for each sample in $U(D_l(k))$, it is performed the upgrade of $Q_k(\boldsymbol{s(k)},\boldsymbol{a(k)},\theta)$ and, once it has been performed for the J samples, a new minibatch of experiences is selected. The upgrade of $Q_k(\boldsymbol{s(k)},\boldsymbol{a(k)},\theta)$ for the experience e_j=< $\boldsymbol{s_{j-1}(k)}$, $\boldsymbol{a_{j-1}(k)}$, $r_j(k)$, $\boldsymbol{s_j(k)}$ >, the following mean squared error (MSE) loss function $L(\theta)$ is considered:

$$L(\theta)= E[(r_j(k)+\gamma\, max_{\boldsymbol{a'(k)}} Q_k(\boldsymbol{s_j(k)},\boldsymbol{a'(k)},\bar{\theta}) - Q_k(\boldsymbol{s_{j-1}(k)},\boldsymbol{a_{j-1}(k)},\theta))^2] \quad (7)$$

Then, the gradient descent of $L(\theta)$, $\nabla L(\theta)$, is considered to update θ, which is obtained by differentiating $L(\theta)$ with respect to θ, resulting in the following expression:

$$\nabla L(\theta)=E[(r_j(k)+\gamma\, max_{\boldsymbol{a'(k)}} Q_k(\boldsymbol{s_j(k)},\boldsymbol{a'(k)},\bar{\theta})- Q_k(\boldsymbol{s_{j-1}(k)},\boldsymbol{a_{j-1}(k)},\theta))\nabla_\theta Q_k(\boldsymbol{s_{j-1}(k)},\boldsymbol{a_{j-1}(k)},\theta)] \quad (8)$$

Then, the weights in the $Q_k(\boldsymbol{s(k)},\boldsymbol{a(k)},\theta)$ network are upgraded according to:

$$\theta \rightarrow \theta+\tau\nabla L(\theta) \quad (9)$$

where τ is the learning rate. Finally, the obtained $Q_k(\boldsymbol{s(k)}, \boldsymbol{a(k)'},\theta)$ is used to update the ε-greedy policy $\pi(k)$. Moreover, during the update of weights θ, the DQN agent has a counter m of the number of performed updates and, when $m=M$, the weights in the target DNN are updated, i.e. $\bar{\theta}=\theta$.

e) Multi-agent operation

Although the triggering of actions by each DQN agent is performed independently of others, the different DQN agents operate in a collaborative and coordinated manner.

On the one hand, a collaborative reward is selected, since the reward definition in (5) for tenant k considers the SLA satisfaction factor of both tenant k and the other tenants in the system in order to avoid selfish actions that would worsen the other tenants. The objective of this is to potentiate those actions that benefit both tenant k and the other tenants.

On the other hand, the triggering of actions and storing of experiences of the different DQN agents is performed synchronously in every time step. Then, once the actions of all agents are obtained, it is necessary to validate that the selected actions in each cell n do not lead to exceeding the total cell capacity (i.e. to avoid that $\sum_{k=1}^{K} \alpha_t(k,n) > 1$). In that case, actions of tenants aiming at decreasing or maintaining the capacity share in the cell, i.e., $a_t(k,n) \in \{-\Delta, 0\}$, are firstly applied and the resulting cell capacity availability $\alpha_{ava,t}(n)$ is computed. If there is no available capacity (i.e., $\alpha_{ava,t}(n)=0$), the actions of tenants willing to increase its capacity in the cell are not applied. In turn, when there is available capacity (i.e., $\alpha_{ava,t}(n)>0$), $\alpha_{ava,t}(n)$ is distributed among those tenants with $a_t(k,n)= \Delta$ proportionally to their $SAGBR(k)$ value, as long as they are not already provided with more than $SAGBR(k)$.

f) Evaluation methodology

The proposed approach will be evaluated by means of system-level simulations using TensorFlow for implementing the DQN agent. The performance assessment of the model will be evaluated in terms of different KPIs, such as the throughput per cell/tenant, the assigned physical resources per cell/tenant, the SLA satisfaction or the utilisation of physical resources.

4.5 Optimal network access problem

4.5.1 Problem statement

In optimal network access, optimal communication resources *matching and allocation* are combined together to satisfy diverse requirements from users and services, in terms of latency and QoS, and maximize utilization of networking and computing resources. Communication resource matching aims to generate optimal policies to match or to associate User Equipment (UEs) with Access Points (APs) and/or Base Stations

(BSs, e.g., 5G gNB) in order to satisfy requirements of delay and throughput (i.e., KPIs). Communication resource placement allocates spectrum, channel, and relevant AP/BS access resources to UEs to satisfy the given users, services, and network requirements.

The network access problem can be considered as a combination of two NP-hard problems, including many-to-many matching and bin-packing. The 5G network scenario impose extra challenges with additional challenges due to the decentralization of control, the sliceability of 5G networks, the complexity and heterogeneity of the network infrastructure (e.g., 4G, 5G, Wi-Fi, LiFi), the mobility of UEs, and dynamic changes in traffic and performance conditions in radio and networks links. Hence, combining, studying and solving the network access problem with multi-objective analysis and with machine learning based approaches are essential to enable multi-tenancy in convergent Cellular, Wi-Fi and LiFi based Edge Infrastructure proposed in 5G-CLARITY. One suitable approach considered in our preliminary study is the modelling of a DRL based approach to provide a distributed solution without requiring global and accurate network information. Besides, DRL can reduce the requirement of computational resources especially in a complex network with a large state space. Our study aims to deploy the proposed solution in a realistic testbed to emulate different types of use cases.

4.5.2 State of the art

Several studies deal with sub-problems of the network access problem. In this section we introduce some relevant works to our study. In the first stage, the simple scenario with a single UE is researched. In [124], a policy learning approach for the optimal selection of channels is presented to minimize the interferences of the UE in a wireless sensor network. Deep Q Learning (DQL) with experience replay algorithm is applied to solve this multichannel access problem. The DQL algorithm generates positive or negative rewards to the UE selection to apply the policy learning. As a result, to get the most long-term reward, the UE keeps adjusting the policy until it is optimal. The simulation results show that the DQL approach, which can provide a near-optimal solution without any knowledge of the system dynamics, is adept in policy definition in the network access problem. However, lacking the retraining mechanism, this strategy cannot adapt to the dynamics of the 5G network. Then, [125] introduces a threshold for the long-term rewards to address the changes in the network states not addressed by the previous study. As a result, when the network states change, the decrease of the reward can trigger the retraining of the DQL model to adapt the learning process to the changes.

Furthermore, the dynamic spectrum access problem for multiple UEs sharing multiple channels is considered. The work in [126] provides a solution for the network access problem in the multi-sensor scenario in which a sensor acts as a relay to collect and buffers the traffic from all the other sensors by simplifying the solution of the problem and solving it with a DQL model. However, only one relay is considered in this paper, and scenarios with more are than one relays require further research. Moreover, [127] presents a study of the dynamic spectrum access problem for multiple UEs sharing multiple channels assuming an attempt probability for each UE to select a channel and transmit during every timeslot to avoid network congestion. To solve the problem, the DQN model located in the BS learns the policy for all offline UEs to decide the pairing relationship between the UEs, the channels, and the corresponding attempt probability of each pair. The study in [127] also proposes the usage of Double Deep Q Network (DDQN) with a duelling DQN to mitigate the over-estimation issue of Q-learning. Simulation results confirm improvements in the throughput. However, learning the policy in a centralized manner may increase the overhead, especially in a dynamic network.

Another relevant study in [128] developed a model-based DDQN to optimise the one-to-one matching between UEs and channels for the maximum long-term data rate in a HetNet with multiple UEs and multi-scale BSs. The simulation results have shown the advantages of the proposed method in convergence speed and the system capacity. However, only the one-to-one matching is considered here, but in fact each channel

can be shared with many UEs and one UE can occupy several channels. Therefore, the DDQN-based method for many-to-many matching between the UEs and the channels is an open issue waiting for further research.

4.5.3 5G-CLARITY initial design

Figure 4.7 represents the designed private network with N UEs, one gNB (5G BS) with M channels, one Wi-Fi AP with K channels and one LiFi AP with N channels randomly distributed in the private network. Hence, there are M+K+N channels provided to serve N UEs. In this section, we are focused on a real-time network access problem in which a UE requires to transmit, and expects services from the BS/AP as soon as possible, as long as network resources are enough. The response time between the task generation and beginning of the transmission is mainly composed of two components: *i)* The time spent on waiting for a suitable time slot to transmit, which is in the order of milliseconds or seconds since the time duration of each frame is 10ms in 5GNR; *ii)* the running time of the ML algorithm for choosing the action, which might take several seconds. Therefore, the time scale for network access procedure is in the order of milliseconds or seconds and can be approximated as real-time.

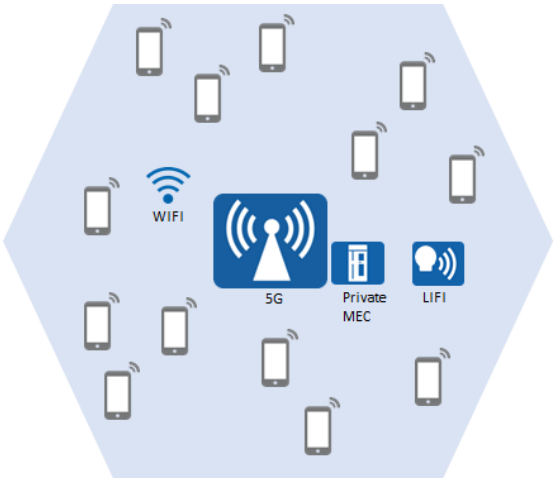


Figure 4.7 Private network scenario with gNb, Wi-Fi AP, LiFi AP and N UEs.

To build a model to solve this problem, historical data needs to be collected from a testbed or the NS3 network simulator periodically. The historical data that can be collected from the telemetry collector include: the transmission rate of each channel, noise power of each channel, bandwidth of each channel, transmit power of each AP, GPS location of each UE, location of each AP, throughput of each UE, and the file size. Meanwhile, the data that needs to be collected from the slice manager include the maximum tolerable delay of each service, and the maximum tolerable SINR of each service.

The output of the model would be a choice that each UE picks out from M+K+N channels. Consequently, an access request from the UEs will be sent to the 5G Access and Mobility Management Function (AMF) which manages the attachment of the terminals to the gNB in order to request the proper available channel and access. The attempt probability will be one of the outputs impacting whether a UE transmits or not to avoid network congestion.

In our preliminary evaluation, we consider DRL a promising technique to be developed to solve the network access problem. Both the UE-centric and BS-centric methods will be researched.

The logic of UE-centric method is presented in Figure 4.8. Every UE with an embedded AI model can choose an AP to connect. Furthermore, each UE updates the AI model based on the information collected independently. To be more precise, when a UE wants to execute a compute task, the AI model within the UE will works to choose one AP or channel based on the global network state. Then, every decision is evaluated

and assigned a reward based on a reward function, which will be designed using the traditional game theory. All the UEs share and buffer the information about this transmission and the change of the network state. Based on the historical transmission data, all the UEs update the weights of the ML algorithm. This method can adapt to a dynamic environment better but may lower the efficiency.

Figure 4.9 describes the BS-centric method. The BS collects historical data to train and retrain the AI model when the environment characteristics change significantly or after a preset period. Then, the BS shares the weights of the trained model with each UE. When a UE has a task to deal with, the shared AI model will compute and get the choice of the action. All the data will be sent to the BS and wait for the next retraining of the model. This method avoids using game theory to get the optimal solution and obtain high efficiency.

Our planned evaluation methodology can be summarized as follows. Firstly, the method will be tested by the data generated from the NS3 network simulator or further from the testbed. Secondly, the adaptability of the method should be evaluated in different network environments, topologies, and requirements. Thirdly, the performance of the method can be evaluated by observing different ML metrics such as the learning rate. Finally, the method will be compared with the other machine learning algorithms or traditional network access algorithms based on the selected KPIs.

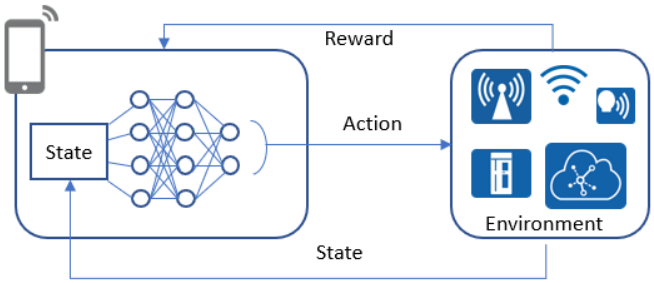


Figure 4.8 Logic diagram of the UE-centric method.

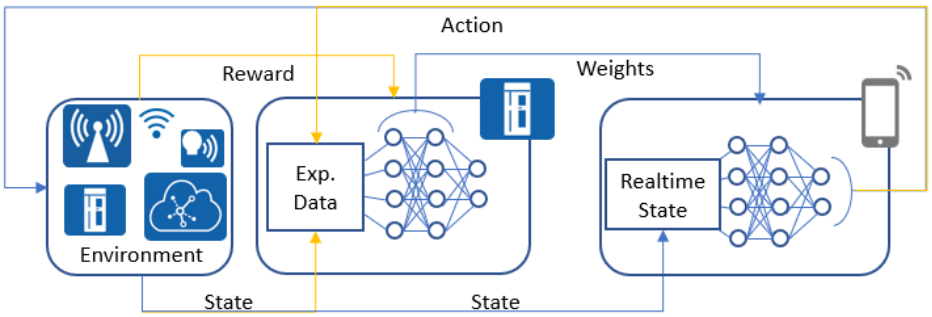


Figure 4.9 Logic diagram of the BS-centric method.

4.6 Optimal compute offloading

4.6.1 Problem statement

With the rapid development of massive Internet of Things (mIoT) at an unprecedented speed, the corresponding deployments may include thousands or millions of heterogeneous and resource-limited devices connecting to the network and collecting massive data, which can trigger bottlenecks in the network and compute infrastructure. In addition, these devices often require connectivity, storage, and processing

capacity at the edge, to meet the performance and resiliency requirements of new services such as Industry 4.0.

To address these challenges, network traffic must be offloaded to the complimentary AP (e.g., Wi-Fi, LiFi), as addressed by the algorithm presented in Section 4.5, and the computational tasks and data storage associated with the services should be offloaded to nearby MEC servers and/or distributed between the BSs, APs, and even neighbouring devices. However, optimal compute offloading will require dynamic definition and update of offloading policies on each device (e.g., UE, IoT devices), network element, and computing elements at the edge by taking into consideration multiple optimization variables, requirements, and network and computing constraints as well as large and dynamic volume of data and connections generated by mIoT.

Hence, we define the *Optimal Compute Offloading (OCO)* problem as the optimal policy to offload the appropriate amount of workload to appropriated available resources distributed through the edge of the network. In our problem the optimal offloading policy definition and update are based on multiple and dynamic network states using the services requirements in terms of processing and storage, the location, throughput/channels bandwidth, level of power of the signal, and available computing and storage resources in devices UEs/IoT, in radio nodes (Wi-Fi, LiFi, LTE, and 5GNR), and in computing nodes located in private/public Edge computing and Cloud computing infrastructure. We note that our defined OCO problem can be seen as a complementary problem to the optimal network selection problem presented in Section 4.5.

Given the complexity and multi-objective nature of the problem as well as the dynamic and heterogeneous infrastructure (i.e., large and dynamic number of network states) required to define and update optimal policies in mIoT/5G infrastructure, the OCO problem is intractable for traditional optimization approaches. As a result, we study the OCO problem as multi-objective optimization problem and propose a Deep Learning based approach to deploy compute workloads in distributed or centralized mIoT and 5G infrastructures.

4.6.2 State of the art

Some relevant studies on Compute Offloading in 5G networks were proposed in the literature focusing on the cost and energy efficiency using AI assisted approaches to address the intractability of the problem in dynamic scenarios and combined with edge/fog computing. The authors in [129] studied the monetary cost reduction by predicting the EU mobility using Deep Q Learning (DQL) algorithm to offload cellular data traffic to the complimentary WLAN. The proposed DQL learns the offloading policy and the convolutional neural network (CNN) predicts the states of UEs when the mobility patterns of UEs are unknown. With the same objective as [129], [130] considers the computation offloading problem in a MEC-enabled network, in which the DQL algorithm assists one central BS to offload the computation tasks of the multiple mobile users to one nearby MEC server in an optimal way.

Furthermore, [131] proposes a Double Deep Q Network (DDQN) algorithm to learn the optimised offloading policy for computation tasks in a more complex scenario with multiple BSs and one shared MEC server. Although the new algorithm can significantly improve offloading performance, it causes a huge waste of energy during the off-peak period. To minimise energy consumption, a new DQL-based algorithm is proposed in [132] to control the activation of the small base stations. Moreover, the work in [133] assumes the resource of multiple MEC servers as a part of the cloud resource to simplify the offloading problem in the network with multiple MEC-enabled BSs. The proposed reinforcement learning algorithm has two layers, the first layer determines an optimal cluster for each offloading task based on a DQL algorithm; and the second layer specifies the physical machine to serve the given task based on Q-learning. Other related studies focusing on computing offloading problem in the edge/fog networks for Industry 4.0 are introduced in [134] and in [135] by considering the offloading from resource-limited private networks to the public networks.

However, no works in the literature considered a comprehensive study of OCO problem as multi-objective optimization problem with additional complete set of realistic parameters collected from public and private mIoT and 5G testbed infrastructures (e.g., device location, LiFi technology, available private and public compute resources including private MEC nodes and Cloud Computing).

4.6.3 5G-CLARITY initial design

As shown in Figure 4.10, our system model considers several UEs, one gNB with M channels, one Wi-Fi AP with K channel and one LiFi AP with N channels that are randomly distributed in the private network. From the perspective of the computing resource, there is a private MEC in the private network, and a public MEC and a centralized cloud in the public network. Historical data will be generated from a testbed or a simulator (e.g. NS3), which can be made available to the ML model through the 5G-CLARITY telemetry collector. Required data will include:

- GPS Location of UE
- Location of Wi-Fi and LiFi APs
- Data rate of channels
- File size of each task
- Transmit power of each AP
- Throughput of each UE
- Size of computing resources:
    - On-device computing resource
    - Local private computing resource
    - Public MEC computing resource
    - Cloud computing resource

In addition, the maximum tolerable delay of each service, maximum tolerable SINR of each service and the unit price of the computation resource can be derived from 5G-CLARITY Management and Orchestration stratum.

The output of the model will be a decision about the offloading of each computational task. Firstly, for data offloading, the output of the model is whether the data traffic will be transmitted through gNB, or offloaded to Wi-Fi and LiFi APs, or will not be transmitted and wait for the next time slot to get a better choice. Moreover, for the computation offloading, the output will be the decision that the computational task will be sent to the local private MEC computing resource, public MEC computing resource, cloud computing resource, or it should wait for the next time slot to get a better choice. In our preliminary study, we consider a DRL, especially the deep Q network, to solve the offloading problem according to the different levels of network dynamics and the size of the on-device computing resources. As in Section 4.5, both the UE-centric method and the BS-centric method will be researched and compared. For the UE-centric method, the AI model is located in each UE (see Figure 4.8), while the AI model is built in the BS in the BS-centric method (see Figure 4.9).

A similar evaluation methodology as described in Section 4.5 will be used.

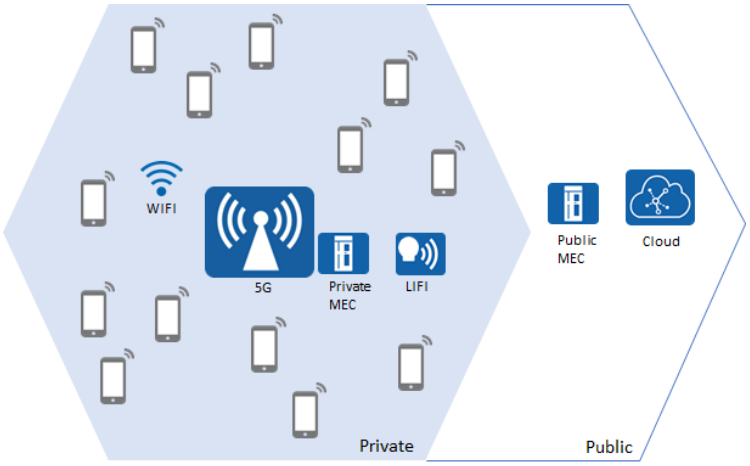


Figure 4.10 Scenario for OCO study.

4.7 Indoor ranging with nLoS awareness

4.7.1 Problem statement

In wireless communication systems, non-Line-of-Sight (nLoS) identification problem can be described as shown in Figure 4.11. Signals travelling between the transmitter and receiver are sometimes obstructed by different objects, e.g., people, walls, furniture and doors. If a signal propagates directly between the transmitter and receiver, it is considered as Line-of-Sight (LoS) condition. On the other hand, if there is no direct path between the transmitter and receiver, it is known as nLoS condition which results in multipath propagation. In particular, knowing whether the link condition is nLoS or LoS can be of great use in several applications such as localization, where nLoS link condition results in a bias in the estimation of the distance between transmitter and receiver.

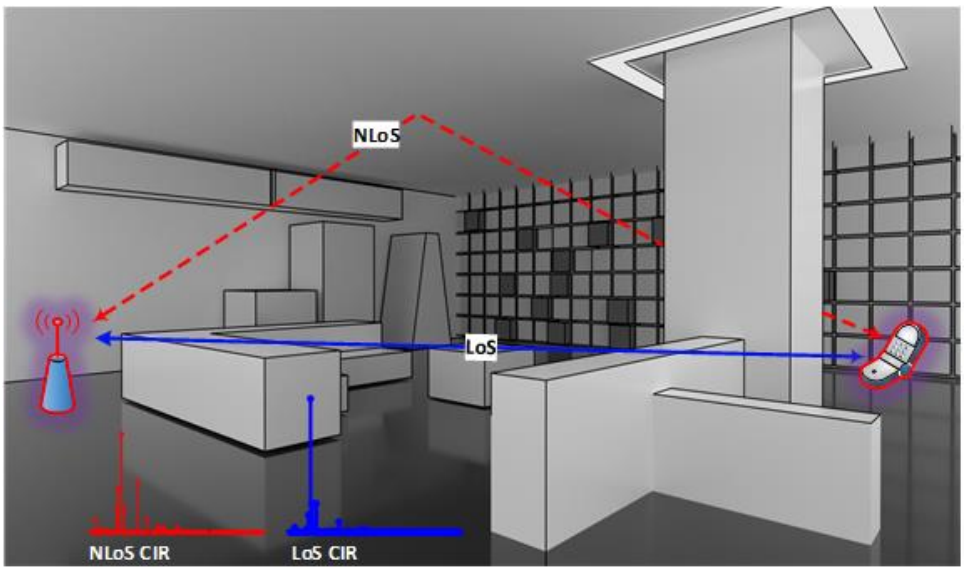


Figure 4.11 nLoS Scenario.

4.7.2 State of the art

Many studies have been conducted to leverage ML approaches to distinguish the LoS and nLoS scenarios.

The algorithm needs first to learn from the training data to generate the classification model. Then the trained classifier is applied to test and recognize the unknown data samples containing nLoS and LoS information. By considering how the training data is represented, these approaches can be divided into two main categories, supervised learning and unsupervised learning.

In the supervised ML, the training data has to be labelled according to number of available classes (at least two classes, LoS and nLoS). In other words, the supervised ML can be also referred to as binary classification. Due to the superb performance of supervised ML approaches, they are popular and used extensively in the literature [136]. In particular, one disadvantage is that the classification accuracy might degrade once indoor conditions or geometry are changed as the training data might need to be updated to generate new classification model. Moreover, the supervised learning approaches require some efforts to maintain, label and validate the data, especially if a large-scale data is needed. Conversely, in unsupervised learning, there is no need for explicit and prior labelling of training data, thereby reducing lots of efforts and time [137]. However, since the unlabelled training data is utilized, the unsupervised ML algorithms are expected to be more complex with intensive processing time. In other words, by using unsupervised learning the complexity (both labour effort and time) of labelling the training data has been moved to the algorithm itself rendering it more computationally complex. The comparison between the above-mentioned machine learning approaches can be summarized in the following Table 4-1.

Table 4-1 Comparison of ML Approaches.

Criteria \ Approach	Supervised Learning	Unsupervised Learning
Training overhead	large	no training phase
Classification accuracy	high	average
Implementation complexity	low	high

Several nLoS identification techniques have been introduced in the literature. nLoS identification Support Vector Machine (SVM) is a supervised ML approach proposed in [138]. It extracts and utilizes features from the Channel State Information (CSI) for nLoS identification based on the Least-Square (LS)-SVM algorithm. While SVM generally has a high computational burden, LS-SVM has overcome this problem by solving the linear equation instead of quadratic programming problem. Moreover, to address nLoS identification, the Random Forest (RF) algorithm is introduced and investigated in [139]. Although the algorithm achieves a superb performance, in terms of both training time and accuracy, it is not yet evaluated in dynamic scenarios, e.g. mobile node. Another CSI-based indoor localization system is AmpN [140] which adopts the Back Propagation (BP) neural network and K-Mean algorithm to achieve real-time LoS/nLoS identification. In [141] the authors present DeepLocNet which is a deep learning-based classifier that utilizes the Received Signal Strength Indicator (RSSI). In general, neural network-based approaches require a great amount of data for training which might not be always available, especially when dynamic environments are dealt with. Finally, an example of unsupervised ML approach is introduced in [142] where the authors applied Expectation Maximization for Gaussian Mixture Models (EM-GMM) to identify LoS and nLoS propagation. While this approach does not need training phase and is of advantage in this sense, it suffers a high complexity rendering its performance limited in practice.

4.7.3 5G-CLARITY initial design

Among all above-mentioned algorithms, SVM-based approaches appear to be more popular due to their ease of generalization and training [143]. Therefore, for our initial design, we rely on SVM to classify the LoS/nLoS links. In the following, we briefly explain the properties of the proposed algorithm. We aim for

designing a real-time SVM ML algorithm which receives the CSI/RSSI of Wi-Fi links from the telemetry collector. Upon reception of a prediction request, it then returns a prediction about the Wi-Fi link condition, i.e. being LoS or nLoS. The steps explained in the sequel are followed to design the algorithm.

Firstly, knowing that SVM is a supervised ML algorithm (i.e. the training data should be labelled), we train the SVM model in an offline manner. To this end, we, in the first step, extract the statistical feature from the data, i.e. CSI/RSSI. The features that can be potentially extracted are as follows:

- Amplitude based: mean, kurtosis, skewness, Rician K-factor.
- Time based: mean access delay, Root Mean Square (RMS) delay spread.

Secondly, in the training phase, the abovementioned features are utilized to obtain the parameters **w** and **b** in the following equation

$$c(\mathbf{x}) = \text{sgn}[\mathbf{w}^T \varphi(\mathbf{x}) + \mathbf{b}],$$

where c(**x**) calculates the class (1 or -1) given the feature vector **x**, sgn[x] is a function which returns 1 if x>0, 0 if x=0, and -1 if x<0. Furthermore, $\varphi(\mathbf{x})$ is the non-linear transformation applied on data to map them into high dimensional feature spaces, consequently rendering them more separable. In general, the SVM, described by the above equation, separates the two classes by maximizing the margin between the two classes while allowing misclassification to be able to handle non-separable data (see Figure 4.12). Furthermore, employing kernels, i.e. $\varphi(\mathbf{x})$, allows for facilitation of the classification in the problems where data are not linearly separable.

Once a link is identified as nLoS, we use the SVM regressor to calculate the distance between the transmitter and the receiver. The principles of the regressor is similar to that of the classifier except that the parameter we are aiming to predict is continuous. Given that, the above-mentioned equation turns into

$$d(\mathbf{x}) = \mathbf{w}^T \varphi(\mathbf{x}) + \mathbf{b},$$

where d(**x**) is the estimated distance.

Finally, evaluation of the algorithm is carried out using simulations in Python environment with the aid of a dataset collected in a real-world scenario. This consists of

- Evaluating the accuracy of prediction when different subsets of features are used.
- Evaluating the speed of training and prediction.

Moreover, in the second phase, the algorithm is evaluated in practice by receiving real-time data from a real indoor environment communication link.

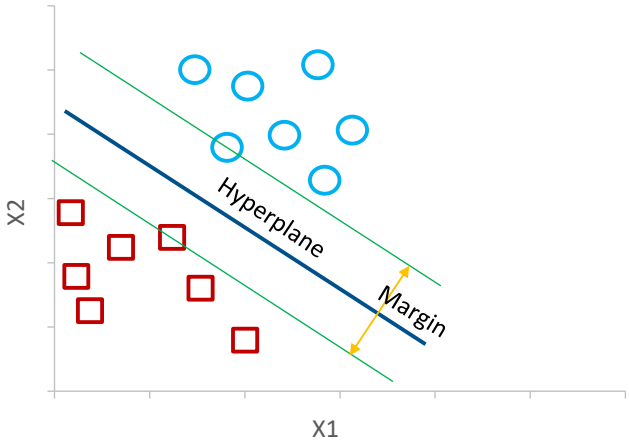


Figure 4.12: Data classification using SVM algorithm.

4.8 Resource provisioning in a multi-technology RAN

4.8.1 Problem statement

Network slicing allows infrastructure providers to partition their physical network into several logical self-contained networks or slices, which are orchestrated in different ways depending on the specific service requirements. For private operators, these network slices can be used to separate their own services or can be offered to service providers and public operators (acting as tenants) through the instantiation of multiple slices. Whatever the deployment scenario is, network slices need to be provisioned with enough networking and computational resources, so that the service quality of such slices is preserved. However, resource provisioning becomes a complicated task, especially in private scenarios as they usually rely on non-3GPP access technologies such as Wi-Fi or LiFi in addition to the pure 3GPP access.

To take advantage of the scenario with multiple RATs, the 3GPP has defined the AT3S function, which stands for Access Traffic Steering, Switching and Splitting [54]. This function implements a set of rules that allow the operator to specify different policies according to its business and technical goals. The AT3S function also adds additional complexity to the problem of slice provisioning in the RAN, since the amount of allocated resources depends on such policies. For example, the traffic of a given slice can be duplicated across several RATs or limited through one RAT, e.g. by establishing a percentage value or routing factor.

The proposed AI-driven 5G-CLARITY algorithm will help the slice manager function manage the radio resource provisioning of different RAN slices in a time-sequential manner. Such slices coexist in a 5G private network scenario with multiple WATs considering AT3S policies. The aim is to improve the performance of the communication services in terms of throughput, latency and reliability.

4.8.2 State of the art

The concept of RAN slicing at different levels is introduced in [48], where each defined level provides a specific degree of granularity in the assignment of radio resources, isolation and customization. The different ways to manage and orchestrate the resources in the RAN is key for RAN slicing provisioning. In this way, the spectrum allocation at the scheduler level is investigated in several works [144][145]. The scope of these works lies on the dynamic resource allocation to cope with the traffic dynamics. At the spectrum planning level, the work in [146] proposes a spectrum planning scheme that maximizes the spectrum utilization.

The existence of multiple WATs, especially in private network scenarios, is also a key aspect in the management of RAN slices. In [147] the general characteristics of Wi-Fi and VLC (or LiFi) are described. The authors explore the existing research activity in this area and demonstrate a practical framework in which both technologies can coexist. In [148], a novel performance analysis is carried out taking the system-level dynamics into account and thus giving a true account on the uplink performance gains of an integrated multi-WAT solution versus legacy approaches. They also advocate for an enabling architecture that embodies the tight interaction between the different WATs and provide 3GPP-compliant simulations to corroborate the mathematical analysis.

With the aim of providing isolation, some works have studied how to implement RAN slicing in Wi-Fi networks using virtualization techniques. In [53], a scheduling algorithm is developed to allocate airtime quotas to a set of virtual interfaces executing on the same or in different physical radios. In [149], a virtualized Wi-Fi network hypervisor based on a time variant radio resource sharing mechanism is presented. Based on the Cloud-RAN paradigm, in [150], Wi-Fi-based DUs are managed through the same CU instance that is used to manage 3GPP nodes. Then, a policy selection mechanism decides on the desired WAT.

In addition to the previous works, some 5G PPP projects have also considered multiple technologies (such as LTE, 5G and Wi-Fi) in network slicing. However, none of them handles with LiFi as a possible WAT. Specifically,

5G GENESIS [151] will use Surrey Platform to demonstrate the effective massive IoT and multimedia communication in multi-WAT environment. In this scenario, 5GNR, LTE-A, Wi-Fi, NB-IoT and LoRa are deployed. To address the resource management issue in disaggregated and heterogeneous RANs, 5G PICTURE adopts a Radio Control Function (RCF) to manage RAN functions according to the tenant's SLA [152]

Regarding the AT3S functionality [92], little research is carried out on this topic so far. In [153], an implementation of multiple-access PDU session to support AT3S in 5G mobile network is presented.

On the other hand, Artificial intelligence (AI) has been successfully applied for resource management in the context of network slicing. In [154], an AI-enabled 5G network architecture is proposed to adjust service configuration and control based on changes in user needs, environmental conditions and business goals. Some interesting AI techniques that have recently been applied to resource allocation of slices are RL [155], deep RL [156][157][158][159], deep learning neural networks [160][86] and evolutionary algorithms [161]. These techniques are particularly effective in handling complicated control problems.

Artificial Intelligence is of such an importance for future 5G networks management that several SDOs and industrial forums are investigating AI and ML techniques to make autonomic decisions by processing huge amounts of data and learning from operations. Some of these SDOs are ITU Focus Group on Machine Learning for Future Networks including 5G (FG-ML5G) [162], established in November 2017 to study network architectures, protocols, interfaces, use cases, algorithms and data formats for the adoption of ML in future networks; and ETSI Industrial Specification Group (ISG) "Experiential Network Intelligence" (ENI) [163], created in February 2017 with the purpose of defining a cognitive network management architecture based on the "observe-orient-decide-act" control model.

The Supplement 55 to ITU-T Y-series Recommendations [164] analyzes a set of use cases of ML in future networks, providing the most relevant requirements related to them. Such use cases are also classified into the following five categories: (i) network slice and other network service-related use cases, (ii) user plane related use cases, (iii) application related use cases, (iv) signaling or management related use cases and (v) security related uses cases. Within the first of the aforementioned categories, radio resource management for network slicing use case is described, focusing on the requirements related to data collection, data storage and the application of the ML output, among others.

4.8.3 5G-CLARITY initial design

In network slicing it is essential to allocate to each slice the needed resources. The mechanism to adapt to load and other system variations in an automatic manner is known as resource elasticity [154]. Specifically, the allocated resources at each time should match the demand as closely and efficiently as possible. There are different time scales at which the system is adapted to traffic changes in a network slice. At low time scales or real-time (around 1 ms), the MAC scheduler assigns radio resources to each UE according to, e.g., buffer status, channel quality, QoS priority, etc. At medium time scales (below 1 s), mechanisms such as the near-real-time RIC [165] provides near-real-time radio resource allocation on a UE/slice basis via fine-grained data collection from different layers of the protocol stack. The periodicity of these actions is such that the near real-time RIC needs to be co-located with the gNB CU. Lastly, at higher time scales (above 1 s), mechanisms such as the non-RT RIC [165] or algorithms residing in the management and orchestration system of the operator network can provision network slices with enough resources to cope with long-term traffic variations and other system events. In this case, there should be a resource reservation for a given slice that is shared by all the UEs attached to the slice and is valid during a certain time interval. Consequently, the management and orchestration system should provide optimized elastic resource provisioning to network slices. Unlike the resource allocation at lower time scales, in non-RT, the allocated resources are not necessary physical resources. For example, they can be expressed as virtual radio resources or defined as quotas.

Specifically, the main network dynamics occurring at a non-RT scale are due to the user behaviour and mobility pattern, resulting in spatio-temporal variations of the traffic demand. For example, there can be differences in the user mobility between working hours and off-working hours in a day. To cope with such dynamics, the network requires a periodical re-distribution of the available resources among the network slices. In a multi-WAT environment, the resource provisioning can also be supported by an efficient load balancing between WATs. In addition, such a re-distribution may also consider key aspects such as the operator's policies related to resource management (e.g. energy saving policies). To carry out the network adaptation, the intended 5G-CLARITY algorithm should follow a closed loop automation strategy, comprising a continuous process that monitors, measures, and evaluates network traffic and then automatically acts to optimize the resource provisioning. The algorithm should perform these tasks periodically, so that the provided network configuration is optimal for the given operator's policies and the average traffic demand in a time period. In each period, the traffic demand may still suffer non-RT variations due to the user behaviour and mobility pattern (e.g. a person walking down a street). The network can react to this kind of changes relying on specific resource allocation mechanisms designed for this purpose. For example, the 3GPP defines the concept of "float quota" in 5GNR to enable dynamic resource allocation. In particular, the network should be able to transfer resources, i.e. Physical Resource Blocks (PRBs), from one slice to another when the former does not need them. The configuration of maximum and minimum quota, as well as possible quota margins, should be performed by the proposed 5G-CLARITY algorithm. Lastly, a resource provisioning algorithm is required to manage the (de-)activation of network slices. Such an algorithm should be eventually executed to re-distribute resources between new slices and other slices in progress. It may also serve to cope with the loss of optimality accumulated by adaptive resource allocation algorithms after successive configuration updates. However, its design is out of the scope of this work. Figure 4.13 illustrates the scope of the proposed 5G-CLARITY algorithm. The targeted parameters defining resource quotas depend on the respective WAT (e.g. number of PRBs, percentage of airtime). For simplicity, only 5GNR parameters are shown in the figure. The frequency of the actions performed by the algorithm are assumed to be in the order of minutes or hours. Consequently, the proposed algorithm can be considered as a non-real time procedure in the 5G-CLARITY system.

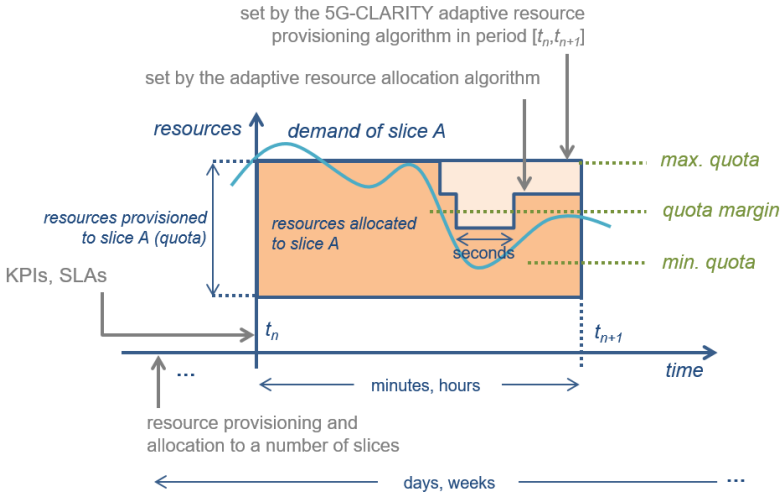


Figure 4.13 The scope of the 5G-CLARITY resource provisioning in a multi-technology RAN.

The 5G-CLARITY wireless network is composed of several WATs and supports the AT3S functionality, involving traffic steering, splitting and switching [166]. According to this, the proposed algorithm handles the resource provisioning between slices and load balancing across different WATs. To this end, the following information should be considered for every network slice:

- Required capacity for guaranteed bit rate (GBR), delay critical GBR or non-GBR (per terminal/slice).

- Maximum packet size.
- Area of service.
- Maximum number of terminals.
- Quality-of-Service (5QI).
- Supported device velocity.
- Isolation (physical or logical).
- Radio resource management (RRM) policies and related parameters (e.g. percentage of resources, strict or float quota).
- AT3S policies and related parameters (e.g. routing factor).

Some of this information can be found in the slice template as part of the SLA and is handled by the Slice Manager. The 3GPP has proposed different AT3S policies to take advantage of the multi-WAT network. From a resource provisioning perspective, the intended algorithm should consider traffic steering and traffic switching policies. Some examples are:

- Traffic steering with best-performance configuration (TST-BP).
- Traffic steering with active-standby configuration (TST-AS).
- Traffic steering with priority-based configuration (TST-PR).
- Traffic splitting with a routing factor (TSP-RF).
- Traffic splitting with redundancy configuration (TSP-RE).
- Traffic splitting with load balancing (TSP-LB).

The traffic switching feature is less relevant for resource provisioning as its main purpose is to provide seamless handover. To re-distribute the resources according to the needs of each slice, the intended algorithm also requires as inputs KPIs and network measurements. This information is periodically gathered by the Telemetry system and accessed by the 5G-CLARITY algorithm for continuous monitoring. Some important metrics are those related to throughput (per user, per slice, per WAT, etc.), resource consumption (e.g. PRB usage) and the number of active users (per slice, per node, etc.). Other relevant information for resource provisioning is related to network configuration, e.g. the transmit power, number of channels and bandwidth of the access nodes.

Based on the network measurements, the intended algorithm should evaluate whether scaling up or down the slice resources or redistributing loads among cells is required to meet the traffic demands. Different network parameters can be used to re-distribute the resources among network slices. In particular, the resource quota determines the amount of resources allocated to each slice. Depending on the respective WAT, this parameter can be defined in terms of number of PRBs, number of channels, percentage of airtime, etc.). The distribution of resources should be made in conjunction with an efficient load balancing among WATs. In the access nodes, parameters such as the transmit power of pilot levels allows to control the traffic loads in a simple way. However, with the introduction of AT3S, a greater number of parameters can be employed to distribute such loads. These are the most representative for each AT3S policy:

- Best performance metric, e.g. received signal level, round-trip-time, etc. (TST-BP)
- Preferred WAT, e.g. 5GNR, Wi-Fi or LiFi (TST-AS)
- Priorities and congestion threshold (TST-PR)
- Routing factor (TSP-RF)
- SINR threshold, below which the policy is triggered (TSP-RE)
- Balance point, e.g. equal load (TSP-LB)

While the TST-x policies allow to control the number of users served by a given WAT, the TSP-x policies allow to distribute the traffic load of each user across different WATs. These inter-WAT load balancing capabilities can be integrated into the resource provisioning procedures in order to maximize the resource usage efficiency.

Figure 4.14 shows the block diagram of the proposed 5G-CLARITY algorithm from a design perspective. The closed loop automated slice provisioning can be seen as a controller that distributes the resources and traffic loads as a function of the current traffic demands and SLAs. To improve the effectiveness of the method, the re-distribution of resources can be carried out taking into account the high-level operator's policies. In particular, the controller's configuration can be adjusted by an optimizer responsible for driving its operation according to the operator's high-level goals. Consequently, the algorithm can follow an optimizer/controller approach, where the optimization framework can be based on deep reinforcement learning. This technique focuses on how to interact with the environment by trying alternative actions and reinforcing the actions producing more benefit in the long run. The resource provisioning controller should be composed of a set of rules and parameters representing the knowledge base. The operation would be as follows: given an evaluation of the performance indicators and the SLA parameters at a certain time, the controller should adjust the radio parameters (e.g. resource quotas) based on its knowledge base. Then, the optimizer should modify the controller behavior in order to meet the operator's business intents. Such business intents or high-level goals should be translated to a mathematical expression called the reward, which can be composed of different metrics gathered from the Telemetry system. Given an evaluation of the reward, the optimizer should update the learned function and adjust the controller behavior (i.e. rules, parameters, etc.) based on the learned function.

Training in reinforcement learning involves an agent interacting with its environment. The proposed algorithm should follow a model-free approach where the agent learns directly from episodes of experience in the live network. However, to speed up the training process, the intended method will be trained with a RAN simulator. The simulated scenario should include a multi-WAT multi-slice network. To support the offline training, the RAN simulator should be computationally efficient. The RAN simulator and the intended ML-based algorithm will be implemented using MATLAB since this tool is appropriate for mathematical programming and computation with matrix algebra. However, it may also be considered the implementation of the algorithm using general-purpose programming language such as Python through libraries that are specifically designed for efficient computation of ML algorithms. In particular, an interesting Python library related to reinforcement learning is KerasRL [167], which implements deep reinforcement learning algorithms and works with OpenAI Gym toolkit. Another library is Tensorforce [168], which also includes an open-source deep reinforcement learning framework and is built on top of Google's TensorFlow framework.

The intended algorithm will be evaluated using a RAN simulator and, if applicable, it will be implemented and evaluated in the 5G-CLARITY pilots. The same performance indicators used by the algorithm, but not limited to them, could be used for evaluation purposes. The performance of the method can be compared to static approaches of resource provisioning. Some reinforcement learning parameters of the intended algorithm could be evaluated, such as the discount rate that determines the relative importance of future rewards and the learning rate. In addition, different behaviours of the algorithm with a particular trade-off between exploration and exploitation as well as the consideration of different business goals could be evaluated.

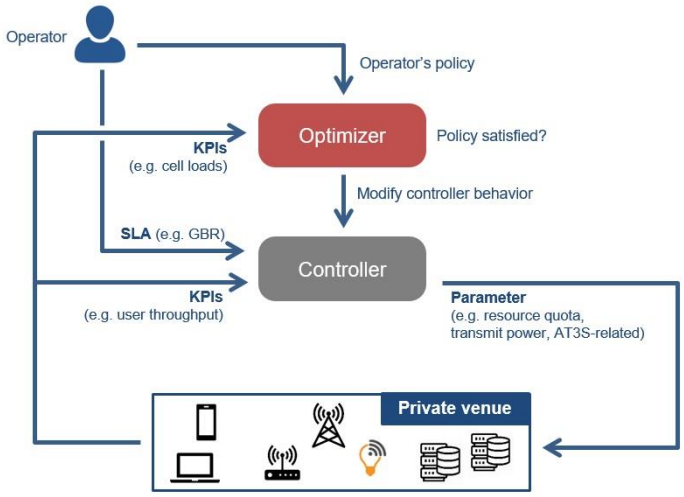


Figure 4.14 Block diagram of the 5G-CLARITY resource provisioning in a multi-technology RAN.

4.9 Dynamic transport network setup and computing resources provisioning

4.9.1 Problem statement

Main enablers of network softwarization are NFV and SDN frameworks. Allowing network operators to automate the resource provisioning and configuration of their networks to meet the required E2E (E2E) performance continuously with agility and in a cost-effective way. Three of the management tasks that impact most on the Quality of Service offered by a softwarized network service (SNS) are the following:

- SNS autoscaling: it refers to the process of deciding the amount of computing resources (e.g., CPU cores, RAM, disk) to be allocated to constituent Virtual Network Function of the SNS so that the SNS target processing performance levels are fulfilled given the foreseen traffic load.
- SNS embedding: it refers to the process of deciding the specific physical resources, i.e., specific (e.g., servers, in which the constituent VNFs instances of the SNS are deployed as either hosted inside virtual machines or OS containers. This process might be subject to meet some constraints such as a bound on the propagation delay and affinity (e.g., two VNFs must run on the same PM).
- Transport network (TN) optimization: it refers to the process of taking the following decisions:
    - The path(s) connecting every pair of VNFs source and destination, i.e., the set of bridges to transport the streams between every pair of end stations. There might be required multiple disjoint paths to assure the reliability constraint imposed by the services. Please note that here we assume a L2 network technology to provide connectivity among network functions. This decision depends on the objectives and constraints indicated below.
    - The configuration of the flow control mechanisms at every output port of the involved network devices. The flow control refers to how the frames are handled at every bridge output port for a given traffic class. This process involves the configuration of the output ports operation to ensure the QoS requisites of the streams. For instance, considering networks with QoS support, the priority for each traffic class must be decided.
    - The network resources (e.g., link capacities) to be allocated to each stream to guarantee its QoS requisites.

The above three processes are coupled as the distribution of the E2E performance budgets among them

might significantly affect the degree of optimality achieved. By way of illustration, the more stringent the SNS response time constraint is, the higher the amount of computational resources allocated to the SNS will be. Then, if our objective is to minimize the required amount of computational resources, it is desirable to assign a lenient delay budget to the SNS processing time. On the other hand, severe performance constraints for the SNS embedding and transport network optimization make it challenging to find feasible embedding and TN configuration solutions.

Here we address the optimization of the three processes listed above in a coordinated way for the SNSs of the different network slices running in the 5G-CLARITY ecosystem. The underlying infrastructure considered consists of two clouds (e.g., RAN cluster and edge cluster) interconnected through an Ethernet-based Time-Sensitive Networking (TSN) network. Here, we consider a multi-objective optimization problem to find a proper balance between the flow acceptance ratio and energy consumption. This problem is subject to the following constraints:

- The assurance of the QoS requirements for the distinct traffic types to be supported. Specifically, the QoS requisites considered are a minimum throughput, E2E maximum delay, and jitter budgets, a maximum packet loss, and a minimum reliability level.
- Technology constraints such as the available physical resources at the different servers within the clouds, the capacity of the TN links, and the available buffer size at each TN bridges' output por.

4.9.2 State of the art

There is a vast literature leveraging Machine-Learning (ML) techniques for dealing with the automation of the daily task management in cloud computing [169]. Typical management tasks are workload prediction, dynamic application, and services scaling (also known as resource provisioning), network functions placement, and load balancing. By way of illustration, in [170] and [171], ML techniques are used for traffic workload forecasting for the resource provisioning of a Network Slicing Orchestration System and elastic Cloud services, respectively. The predicted workload is used to fed queueing models that translate it into required resources given the service response time specified in the Service Level Agreement (SLA). There are also proposals for the network services auto-scaling entirely based on machine learning [172][173]. The authors in [172] employ Q-Learning for VNFs scaling. Whereas in [173], the authors propose a topology-aware graph neural network for the proactive scaling of chains of VNFs.

Regarding the flow allocation, fewer works are addressing the problem using ML techniques. There are two pioneer works on applying machine learning techniques for tackling the flow allocation problem in Time-Sensitive Networking (TSN) [174] and [175]. Indeed, TSN is a candidate scenario that can benefit from ML. That is because of the large number of possible configurations offered by TSN networks. Both works [174] and [175] put stress on the importance of carrying out a validation process of the ML decisions via worst-case performance models. Otherwise, the solution is not fully reliable. In the TSN context, a configuration for a set of flows is valid when it yields the fulfilment of the performance constraints for all the flows. In [174], the authors explore both supervised and unsupervised learning for configuration feasibility check or validation in Time-Aware Shaper (TAS)-based TSN networks. The motivation is that precise schedulability analysis for these networks might be computationally heavy. In this way, a binary classifier could be used during configuration optimization processes. In this scenario, unsupervised ML methods are more practical. That is because the labelling process (feasibility or non-feasibility for each configuration) might be computationally heavy, as mentioned. In [175], the authors propose a Reinforcement Learning (RL)-based solution, dubbed LEARNET, for the flow allocation in Asynchronous Traffic Shaper (ATS)-based TSN networks. They check the feasibility of every action issued by the agent. The output of this check impacts on how the agent is rewarded. Then, the knowledge of the performance models is transferred to the agent, and the solution becomes fully reliable.

Although there are several ML-based proposals for addressing the everyday management tasks in clouds, they focus on a specific problem. Nonetheless, the coordination of the different management tasks using ML seems unexplored yet. Also, most of the solutions proposed in the existing works focus on the delay constraint assurance, but they do not take into account other performance metrics such as reliability, jitter, and packet loss. On the other hand, the application of ML for the flow allocation is still in its infancy. In the context of TSN, two pioneer works propose ML-assisted flow allocation solutions. However, several challenges lie ahead, such as achieving scenario-agnosticism or testing the solutions in more extensive networks. Furthermore, it has been proven that the use of standard optimization techniques such Satisfiability Modulo Theories (SMT) fail to cope with the combinatorial complexity to find feasible configurations for TSN networks.

4.9.3 5G-CLARITY initial design

The algorithm described here aims to find a balance between the flow acceptance ratio and infrastructure energy consumption. To that end, it dynamically adapts to the foreseen traffic loads by setting up the compute quota (e.g., number of virtualized CPUs, number of virtualization containers, and RAM), transport quota (e.g., network capacity and buffer sizes), the embedding of the VNFs for each 5G-CLARITY slice. All of this subject to satisfying the performance constraints imposed by the different traffic types and the available resources at the infrastructure stratum for both domains the TN and computing. The algorithm deals with the workload fluctuations at high time scales, i.e., time frames from several minutes to hours. Then, the algorithm can be regarded as non-RT, i.e., there is no deadline associated with its execution, or the timing constraint to run it is lenient as the frequency of the algorithm's actions is within the same time scale as the workload fluctuations. For instance, the traffic load fluctuations mentioned above might be due to the different activities carried out by an industrial site throughout the day and people's behaviour (e.g., workers or customers). Reactive algorithms might be used to cope with workload fluctuations at short time scales, though that problem is not addressed here. Besides, all the required state information to optimize the network configuration is available before each algorithm run. In this sense, we are considering an offline algorithm to tackle the problem.

The bottom layer of Figure 4.15 sketches the 5G-CLARITY's physical infrastructure considered. There are two clouds, namely RAN Cluster and Edge Cluster. The former might include hardware accelerators like Field Programmable Gate Arrays (FPGAs) and Graphic Processing Units (GPUs) to meet the processing time budgets imposed by the virtualized 5G New Radio functions (e.g., gNB-DU and gNB-CU). Time-Sensitive Networking (TSN) is considered as L2 technology for the transport network (TN). TSN can provide the deterministic QoS required by critical services while enabling their coexistence with non-performance sensitive traffic.

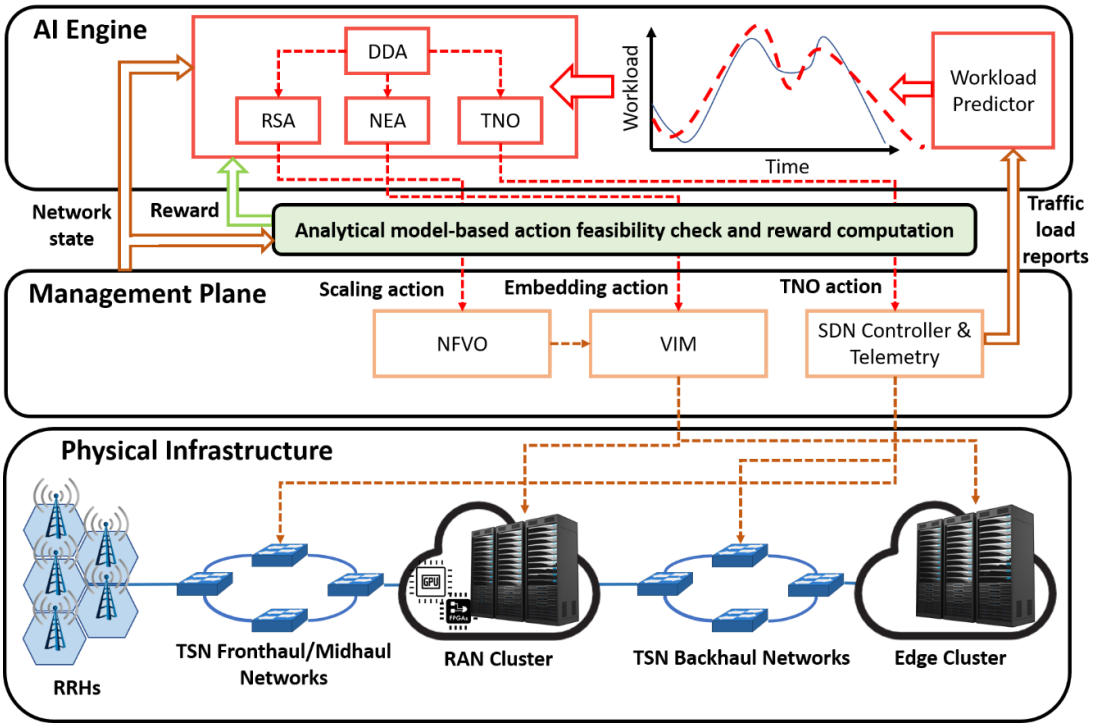


Figure 4.15: Initial design for the ML-based dynamic transport network setup and computing resources provisioning solution.

The proposed solution to address the problem stated in Section 4.9.1 combines ML approaches and analytical model-based techniques. ML is used to deal with the complexity of the optimization process, while the analytical models for facilitating and speeding up the training process and validate the feasibility of the actions issued by the ML agents, i.e., checking whether a given action meets the E2E performance requirements. There are some performance metrics, such as the E2E performance metrics that are difficult to measure on-live from the network. The reason for that is that the worst-case network performance is given by corner cases, which are unlikely to happen. In this regard, analytical models are a practical solution to predict the worst-case performance of the network quickly and with low complexity, thus easing the training process.

On the other hand, the validity of the actions issued by the agents is uncertain. Then, the analytical models are the cornerstone to filter the unfeasible actions (i.e., those that do not guarantee the QoS requisites of the distinct traffic classes) and ensure the full reliability of the decisions taken. On the other hand, ML is required for handling the complexity of the optimization process as analytical-based optimization techniques struggle to cope with it. For instance, it might take several hours to find a feasible solution for a medium-scale TSN network using SMTs

The initial solution consists of a multi-agent architecture, where there are a set of ML agents specialized in the different decision processes (see Figure 4.15). In principle, deep reinforcement learning (RL) is considered as ML technique to implement the agents. There is an RL agent per each decision process listed in Section 4.9.1, namely, resource scaling agent (RSA), network embedding agent (NEA), and TN optimizer (TNO). Besides, there is a delay distribution agent (DDA) responsible for distributing the E2E delay and jitter budgets between the three decision processes to realize their coordination. The RSA, NEA, and TNO are fed with the DDA outputs and other inputs of interest such as:

- Nodes and links time-to-failures.
- Temporal profile demand per 5G-CLARITY slice, per 5QI, and per source/destination pair.
- Traffic characteristics per 5QI (e.g., moments for the data rate process).

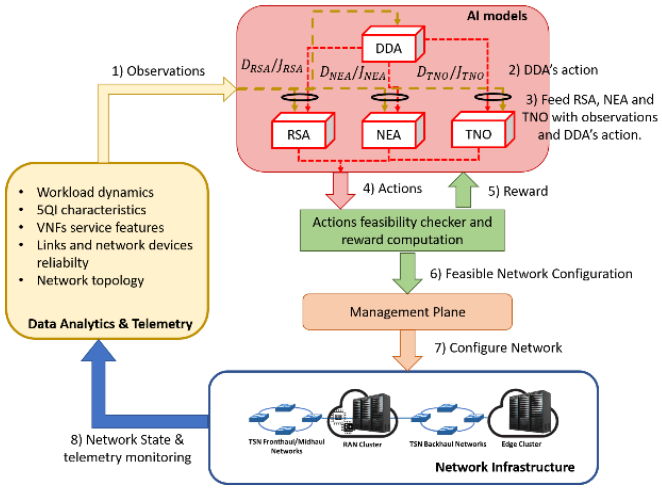
- Processing time distributions for every VNF/Service app.
- Network topology.

The algorithm running at the AI engine obtains most of the above information through the management plane (see Figure 4.15). The foreseen temporal traffic load profiles are provided by an auxiliary ML-based workload predictor, which, in turn, collects timestamped samples of the workload from the telemetry system to learn them.

The agents issue their respective actions for allocating the compute and transport resources, embed the virtualization containers, and configure the TSN TN. These actions are filtered through the use of analytical performance models to check their feasibility. Furthermore, the same models assist the computation of the reward for the agents. If the actions are valid, they are forwarded to the respective management plane entities to be applied to the network, and the agents are positively rewarded. Otherwise, they are rejected, and the agents must search for an alternative configuration. The outputs of each agent are listed below:

- RSA: This agent's action by this agent is forwarded to the NFVO and later applied by the VIM.
  - The compute resource quotas for every VNF of each 5G-CLARITY slice.
- NEA: The action issued by this agent is sent to and applied by the VIM.
  - The mapping between VNF instances and the physical machines or servers.
- TNO: The actions issued by this agent are sent to the SDN controller and later applied by the TSN Central Network Controller (CNC).
  - The translation of the 5QIs into IEEE 802.1Q traffic classes for each 5G-CLARITY slice.
  - Aggregated transport resources (e.g., link capacities and buffer sizes) per traffic class and per 5G-CLARITY slice.
  - Paths allocated for each 5G-CLARITY slice.
  - Output port configuration of the TSN bridges, e.g., gate control list and time windows size.

For the initial training process, we will employ a simulator of a 5G SNPN with the infrastructure shown in Figure 4.15 and synthetic temporal traffic profiles resembling those expected in 5G private network scenarios similar to 5G-CLARITY UC-1. As pointed out previously, the analytical performance models will assist the training process in order to determine the worst-case performance metrics given a workload and a network configuration. Figure 4.16 shows and summarizes the workflow of the initial solution for the dynamic transport network setup and computing resources provisioning.



1) The DDA is fed with the required data analytics and telemetry.
2) The DDA issues its action to distribute the e2e jitter/delay budgets among the RSA, NEA, and TNO models. The AFC validates this action, and the DDA is rewarded.
3) The DDA's output and the required data analytics & telemetry are gathered and provided to the RSA, NEA, and TNO agents.
4) The RSA, NEA, and TNO generate their respective actions whose satisfiability is checked by the AFC.
5) If the RSA, NEA, and TNO actions are valid, all the agents are positively rewarded.
6) The AFC forwards the satisfiable network configuration to the management plane.
7) The management plane applies the configuration in the network.
8) The telemetry system keeps monitoring the network state and metrics of interest until the algorithm is rerun.

Figure 4.16: Workflow of the AI-assisted solution for the dynamic transport network and computing resources provisioning.

Finally, the experimentation and evaluation process of the solution will consist of the following parts:

- Study on the optimal ML hyper parameters configuration.
- Solution performance assessment. To that end, we will measure through simulation the degree of optimality achieved while checking the target E2E performance requirements are met all the time.
- Solution performance comparison. The most relevant SotA solutions will be implemented and their performance compared against the proposed solution.

4.10 Adaptive AI-based defect-detection in a smart factory

4.10.1 Problem statement

This use case considers a production line in a smart factory such as the one targeted in 5G-CLARITY for the Bosch use case. The production line is composed of machines that may be fixed or mobile (e.g. robots, robot arms, etc.) acting continuously or on-demand on the line. Additional sensors including video cameras are used for real-time monitoring and subsequent intervention by the machines, e.g. to stop the line or to remove a defective product. Such intervention by the machines is typically instructed/commanded by a remote factory worker acting upon real-time data received from the sensors and cameras over a local/private or wide area network (e.g. through a 5G network). The machines and devices in the smart factory are assumed to have capabilities for networking, computing, and storage. The computing capability on the local machines and devices in the factory can support distributed data telemetry and intelligent functionalities locally. Numerous cameras and sensors, with the possibility for some cameras to be on-wheels (e.g. carried by guided vehicles), are continuously monitoring the production line. These cameras and sensors are capable of data storage, fast data analysis, including extracting and capitalizing on the corresponding knowledge in real-time. Running Federated Learning across multiple devices allows fast and accurate data analysis. These functions help in detecting more efficiently and quickly characteristic patterns that allow the recognition of potential defects in the production line.

These local capabilities are leveraged together with additional (more sophisticated but mostly fixed) capabilities available in the E2E infrastructure (e.g., telco edge, distant cloud) connecting the smart factory to the remote digital worker. Detection of a defective piece triggers a remote worker to command an intervention by some machines (e.g., robots or robotic arms) to stop the line or take the defective piece out of the production line to a certain destination. Such immediate intervention implies real-time processing and visualization of geometric features for manufactured parts at the remote worker location. Clearly the sooner a piece is detected as defective and taken out of the production line, the less scrap will be generated. Moreover, the faster the pieces are analysed, more pieces can be produced in each period.

The algorithm used for defect detection typically follows a framework for designing and training deep neural networks (DNN), such as for example the Yolov3 algorithm "you only look once" (YOLO). The choice of the detection algorithm and its AI training model depends on the characteristics and capabilities of the resources deployed for both the training and inference environments. These characteristics may vary dynamically and therefore an adaptation to the algorithm and/or its training model, as well as where it is instantiated, may be required to achieve low latency and energy efficient deployment.

The problem solved here is therefore to drive an adaptive low latency and energy efficient AI-powered defect detection based on the telemetry collected from the various devices and nodes involved in the communication and computing infrastructures as depicted in Figure 4.17.

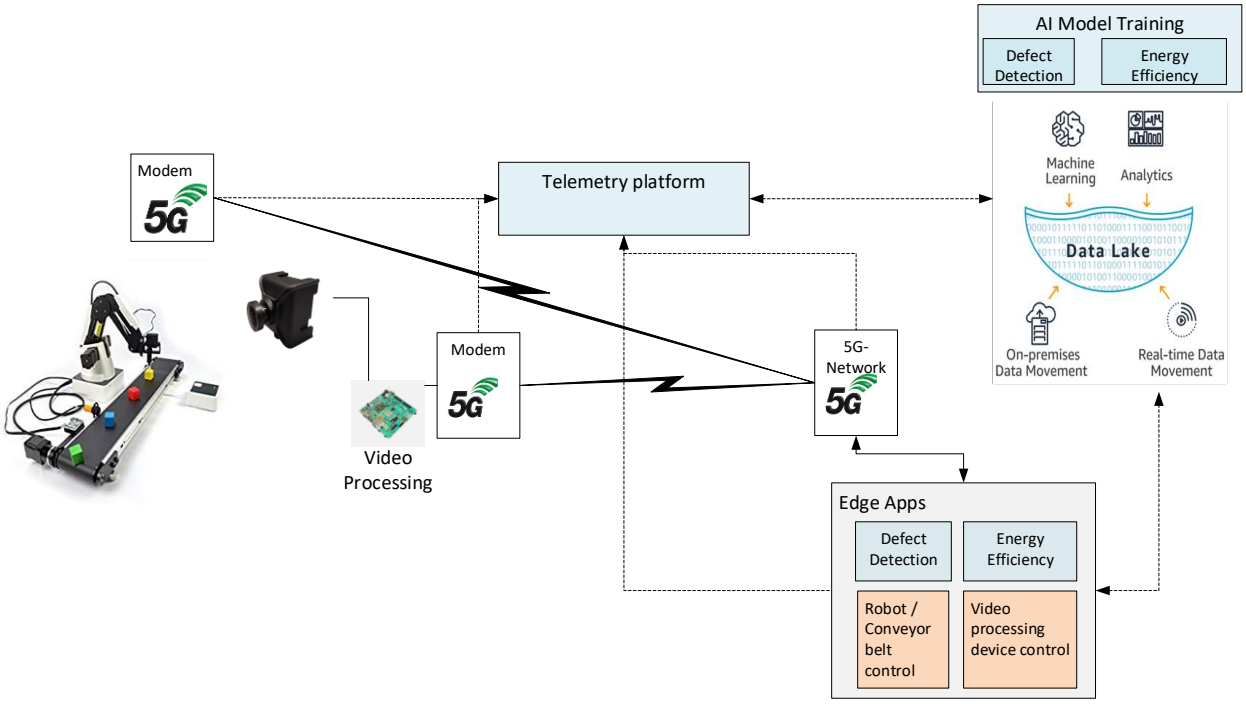


Figure 4.17: Defect-detection in smart factory.

4.10.2 5G-CLARITY initial design

Analytics and intelligence are using telemetry and video-based inference to provide insights from operations that are occurring in the smart factory network. The telemetry may include system state, logs, configurations or other text / binary data. Some of the telemetry may be unstructured and stored as it has been received and other telemetry may be strictly based on standard models. Real-time telemetry can be a collection of measurements or other data in real-time at remote nodes and automatic transmission of the data. The nodes that are transmitting real-time telemetry are counterpart of telemetry deployed either at the edge or at cloud. Real-time telemetry may be consumed also non-RT functions, for instance in data lake scenario.

Video analytics consist of inference performed on the video stream and analysis of those inference results. The video analytics are combined with the telemetry models in order to provide a specific, or overall view of the system allowing defect detection to perform actions that are intelligent in the case when faults occur or when collecting insights from the collected data.

Figure 4.18 depicts the workflow for the 5G-CLARITY initial design for the intelligence-based defect detection module showing the key interactions amongst the different entities:

- Step 1: In a first step, the sensory (including camera) data together with measurements data are sent from various devices to the network via for example one or multiple 5GNR modems.
- Step 2: Some user data and network (including UE modem) telemetry data is passed to an Edge platform. This edge platform includes the defect detection application responsible of the inference. It also includes a telemetry agent from the edge to collect the various telemetry data and aggregate it and pass it further up to the data pool or data lake for processing.
- Step 3: Various telemetry data from the applications, network functions and entities, and edge computing hosts, are passed on to the data lake for storage and pre-processing.
- Step 4: The telemetry data, after pre-processing, is presented in the form of datasets for training or updating the training of different AI models.
- Step 5: The AI model is trained or retrained.

- Step 6: The new AI model, adapted to the actual telemetry data is then obtained.
- Step 7: The output AI model is passed on to the infrastructure orchestration for deployment.
- Step 8: The orchestration and control instantiate the inference algorithm in the edge platform and instructs for any update to the telemetry collection.
- Step 9: The inference is carried out by the defect detection application in the edge, and the result is passed to the control applications of the end devices including sensors, cameras and processors.
- Step 10: The various edge controller applications finally instruct the various devices to execute certain actions including change of configurations.

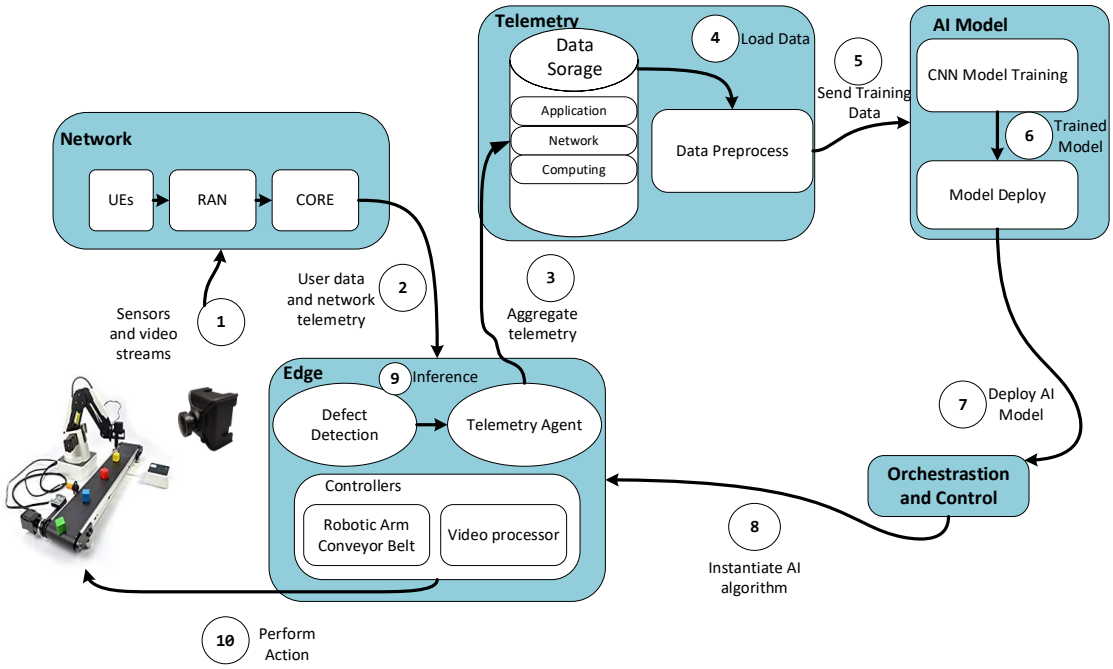


Figure 4.18: Workflow of the 5G-CLARITY solution for defect detection.

5 AI Engine

To benefit from scalable and flexible machine learning in support of network management functions, 5G-CLARITY requires an ML platform that is in line with 5G-CLARITY's (5G-PPP's) service-oriented objectives. The AI engine fills that role, providing a cloud-native platform that hosts ML models as containers for seamless scaling and flexible development and deployment.

5.1 Requirements for an AI/ML engine in 5G-CLARITY

The AI engine component is part of the Intelligence Stratum introduced in Deliverable D2.2 [2], and will be designed according to the following requirements:

1. Use of open source to minimise costs for the 5G-CLARITY research project, avoid vendor lock-in solutions and allow customization.
2. Minimal overhead for installation and long-term maintenance to minimise the effort and corresponding costs towards a feasible research prototype
3. Packaging of ML models as a format that enables seamless deployment into near real-time RIC component operating in the RAN cluster, which supports containerised xApps. The interested reader is referred to 5G-CLARITY D3.1 [3] for an introduction to the near real-time RIC and the xApp framework.
4. Enable experimentation with cutting edge and custom ML algorithms beyond state-of-the-art algorithms
5. Allow for training of ML models to be inside or outside the private venue, e.g. in a public cloud with enough training resources.
6. Enable the lifecycle management of ML models, including instantiation, removal, update of ML models.
7. Provide a registry for ML models for ML service discovery
8. Enable access to ML functions to non-expert private network operators

5.2 State of the art on AI engines

We review in this section the relevant SotA on the introduction of AI functions in mobile networks. To this end we first analyse the efforts being done by standard development organizations in Section 5.2.1 and then look at the ML platforms available in the market in Section 5.2.2

5.2.1 Standardization frameworks

5G networks are facing the challenge of increased complexity due to the increased numbers of configuration parameters, deployment options and corresponding key KPIs such as data rates, latency, reliability, and connection density. AI provides a powerful tool to help operators manage this complexity and better optimize their network performance and automate its operations. For AI tools to work, there is need for a framework to *i)* make available datasets relevant for training and updating the various AI algorithms; *ii)* define the necessary interfaces in the system for integration of the AI algorithms; and *iii)* help derive new system requirements and therefore new design by accounting of the impact AI tools would have on different parts of the system. In this spirit, various industry forums and standardization organizations such as ITU-T, ETSI, 3GPP, O-RAN, have been studying the way to integrate AI into their existing and future networks. Below we present a brief survey of the work carried out in key forums namely ITU-T ML5G Focus Group, ETSI ENI ISG, 3GPP and O-RAN.

5.2.1.1 ITU-T ML5G focus group

The ITU-T ML5G (Machine Learning for 5G and Beyond) Focus Group has been active in drafting technical specifications for ML for future networks, including interfaces, network architectures, protocols, algorithms and data formats. The following specifications have been developed:

- "Architectural framework for machine learning in future networks including IMT-2020" (ITU-T Y.3172, January 2020)
- "Machine learning in future networks including IMT-2020: use cases" (Supplement 55 to Y.3170 Series, October 2019)
- "Framework for evaluating intelligence level of future networks including IMT-2020: use cases" (ITU-T Y.3173, February 2020)
- "Framework for data handling to enable machine learning in future networks including IMT-2020: use cases" (ITU-T Y.3174, February 2020)
- Draft ITU-T Recommendation (progressed in ITU-T SG13): "ML marketplace integration in future networks including IMT-2020"
- "Requirements, architecture and design for machine learning function orchestrator"
- "Serving framework for ML models in future networks including IMT-2020"
- "Machine Learning Sandbox for future networks including IMT-2020: requirements and architecture framework"
- "Machine learning based E2E network slice management and orchestration"
- "Vertical-assisted Network Slicing Based on a Cognitive Framework"

All the above specifications are quite relevant to the AI engine design in 5G-CLARITY and therefore constitute a reference framework to build on. These specifications have also been taken as reference for the intelligence framework in industry forums like O-RAN.

5.2.1.2 ETSI ENI industry specification group

ETSI ENI ISG has been developing specifications for leveraging AI techniques like machine learning and reasoning in the network management system. The ISG has been active since 2017 and is completing its second two-year cycle focused on data and action interoperability. A plan for third two-year cycle is underway and the ISG is anticipated to continue to be active in 2021-2022. The scope of this third cycle is not publicly available at this stage of writing this document. It is also noteworthy a workshop[2] "ENI-Machine Learning in communication networks" organized on 16th March 2020 between two ETSI ISGs, namely ENI and SAI (Securing AI), and ITU-T's Q20/13 and FG ML5G "Machine Language 5th Generation", on AI/ML. This workshop was instrumental in enabling synergies between ETSI ENI and SAI and the ITU-T ML5G which are now set on track to collaborate and complement each other. The following specifications and reports have been published from the Release 2:

- ETSI GS ENI 001 V2.1.1 (2019-09): Use Cases
- ETSI GS ENI 002 V2.1.1 (2019-09): Requirements
- ETSI GR ENI 003 V1.1.1 (2018-05): Context-Aware Policy Management Gap Analysis
- ETSI GR ENI 004 V2.1.1 (2019-10): General Terminology
- ETSI GS ENI 005 V1.1.1 (2019-09): System Architecture
- ETSI GS ENI 006 V2.1.1 (2020-05)V2.1.1 (2020-05)V2.1.1 (2020-05): Proof of Concept (PoC) Framework
- ETSI GR ENI 007 V1.1.1 (2019-11): Definition of Categories for AI Application to Networks

The Use Cases ENI 001, Requirements ENI 002, Terminology ENI 004 and System Architecture ENI 005 are all

[2] https://www.etsi.org/newsroom/blogs/entry/eni-13-progressing-release-2-and-etsi-itu-t-workshop-eni-machine-learning-in-communication-networks

open for the next release and work has started on each specification. The new work items for the next release are also open and under drafting, including:

I. Intent Aware Network Autonomicity in ENI 008

II. Data related mechanisms in ENI 009

III. Evaluation of categories for AI application to Networks in ENI 010

IV. Mapping between ENI architecture and operational systems in ENI 011

V. Reactive In-situ Flow Information Telemetry in ENI 022.

All the above are quite relevant to the AI engine design in 5G-CLARITY in particular the ongoing ENI 008 on intent aware network autonomicity, and ENI 022 on flow information telemetry.

5.2.1.3 3GPP

In its Rel-15 and Rel-16, 3GPP has been specifying the framework to enable data collection and provide analytics to consumers. This included the definition of Network Data Analytics Function (NWDAF) services to support the analytics that are required for QoS Profile Provisioning, Traffic Routing, Future Background Data Transfer, Slice SLA, Performance Improvement and Supervision of mIoT Terminals, Support of Northbound Network Status Exposure and Customizing Mobility Management. Release 17 aims to improve upon the work initiated in Rel-15 and Rel-16 and is currently focusing on:

- **SA2** – R17 SI (SP-190557) Enablers for Network Automation: Data collection for NW automation and Support NW Data Analytics Functions for NW optimizations.
- **RAN** – R17 SI (RP-201304) Further enhancements on data collection: Definition and signaling for interoperable multi-vendor input / output data.
- **SA1** – R18 SI (SP-191040) Application-based AI/ML - model transfer in 5GS: Traffic model for AI model transfer (eMBB) and Traffic model for Remote inference/relearning (uRLLC).

All the above activities in 3GPP Rel-17 are quite relevant to the AI engine and telemetry work in 5G-CLARITY.

5.2.1.4 Open RAN Alliance (O-RAN)

The O-RAN alliance promotes the interoperability of disaggregated RAN solutions and aims to leverage deep learning techniques to embed intelligence in every layer of the RAN architecture [176]. The most noticeable AI-relevant activities in O-RAN are the design of RICs. These include the non-RT and near-RT. The Non-RT control functionality is defined as > 1s and near-Real Time control functions control functions are < 1s. near-RT and non-RT control functions are decoupled in the RIC by O-RAN architecture.

Non-RT functions include service and policy management, RAN analytics and model-training for the near-RT RAN functionality. Trained models and real-time control functions produced in the non-RT RIC are distributed to the near-RT RIC for runtime execution. O-RAN defined A1 is the interface between the non-RT RIC and the near-RT RIC. With the introduction of A1, network management applications in non-RT RIC can receive and act on highly reliable data from the modular CU and DU in a standardized format. O-RAN aims to create next generation RRM with embedded intelligence, while supporting legacy RRM. The enhanced O-RAN RRM targets to resolve operational challenging functions, for instance, per-UE controlled load-balancing, RB management, interference detection and mitigation. In addition, the new intelligent RRM provides new functions leveraging embedded intelligence, such as QoS management, connectivity management and seamless handover control.

Relevant AI interfaces are A1 and E2. A1, as described above, is the interface between non-RT RIC and the near-RT RIC. E2 is the control plane interface between the near-RT RIC and the multi-RAT CU protocol stack

and the underlying RAN DU. Originated from the interface between legacy RRM and RRC in traditional systems, the E2 delivers a standard interface between the near-RT RIC and CU/DU in the context of an O-RAN architecture. While the E2 interface feeds data, including various RAN measurements, to the near-RT RIC to facilitate radio resource management, it is also the interface through which the near-RT RIC may initiate configuration commands directly to CU/DU.

The near-RT RIC can be provided by traditional TEMs or 3rd-party players. While receiving an AI model from non-RT RIC, near-RT RIC will execute the new models (including, but not limited to traffic prediction, mobility track prediction and policy decisions) to change the functional behaviour of the network and applications the network supports.

5.2.2 Existing ML platforms

The current landscape of AI/ML provides an unprecedented level of choice of tools and frameworks for vastly different use cases, from libraries that package specific ML algorithms to complex ML lifecycle pipelines. For a high-level overview, we refer to the Linux Foundation AI Landscape in Figure 5.1. The remainder of this section will highlight and briefly describe the features of selected commercial as well as open source ML platforms. Note that this is not an exhaustive list. We observe that much of the functionality of the discussed ML platforms is offered by other vendors/platforms as well and the choice of ML platform is often dictated by the existing software ecosystem.

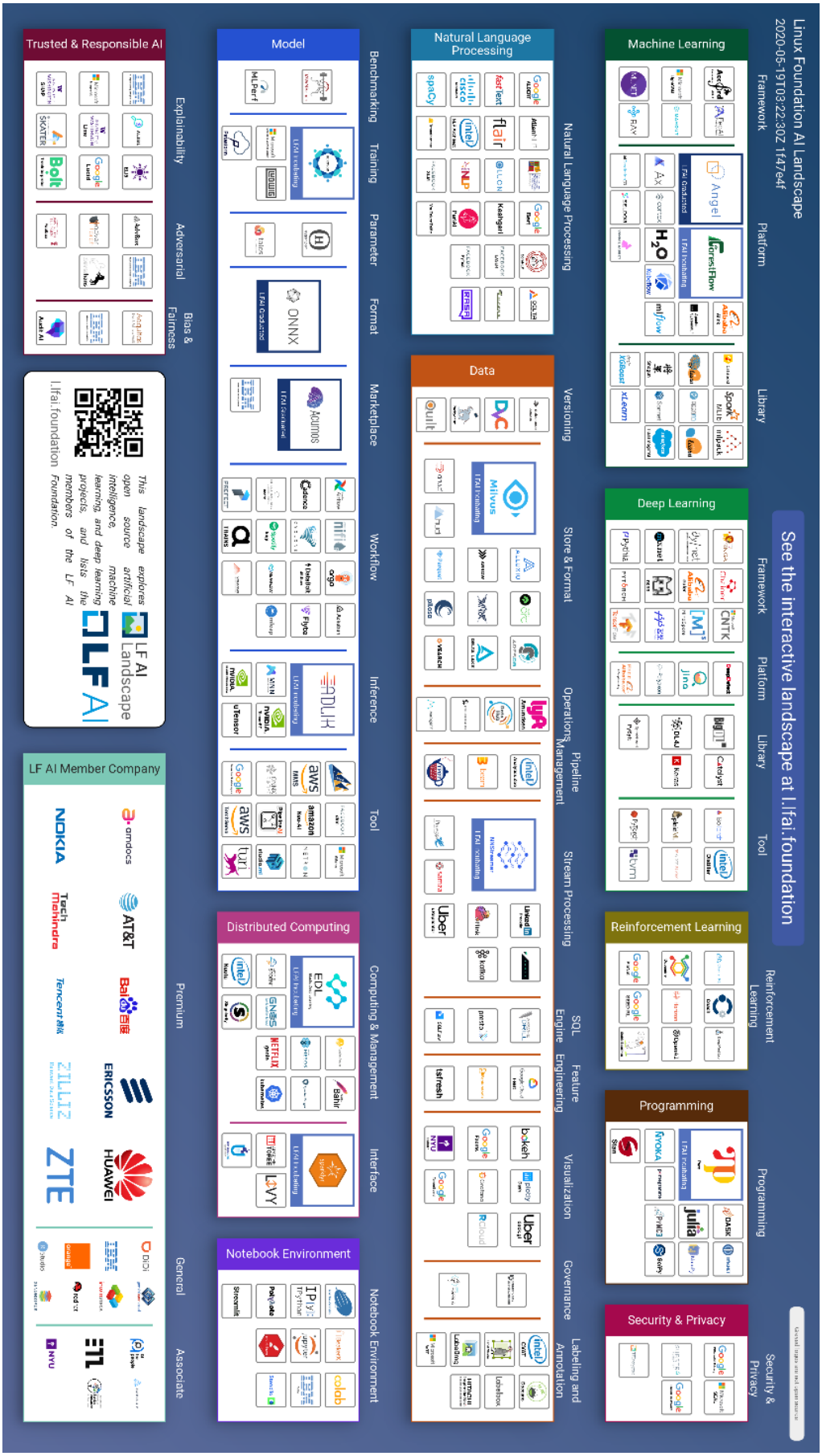


Figure 5.1 AI landscape according to the Linux Foundation (https://landscape.lfai.foundation/).

5.2.2.1 Commercial: amazon AWS SageMaker

SageMaker [177] is Amazon's cloud ML platform. It allows users to build, train, and deploy machine learning models in one cloud service. The goal of SageMaker is to decrease time to production, development effort and cost by supporting the entire ML workflow (and partial automation therein), including:

- Data labelling and preparation (for highly accurate training datasets).
- Algorithm selection (built-in high-performance algorithms, local testing and prototyping).
- Model training, tuning and optimization (in notebooks, one-click training, managed Spot training, automatic model tuning, train once run anywhere, model tracking).
- Deployment and prediction-based actioning (one-click deployment, fully managed hosting with auto scaling, batch prediction, inference pipelines).

This E2E workflow also includes a managed environment for secure and fast model testing and hosting. SageMaker runs on Amazon's cloud service AWS and is therefore a paid AI/ML framework.

5.2.2.2 Commercial: Microsoft Azure machine learning

Similar to Amazon AWS, Microsoft Azure offers a cloud-hosted ML platform that aims to automate many steps in the ML workflow with Azure Machine Learning [178]. Besides ML workflow automation, Azure Machine Learning also offers Designer, a graphical user interface that allows to visually assemble ML pipelines in a fast and simple manner. This way, typical ML pipelines can be created without writing actual code, which makes it easier accessible for a wider range of audience.

A typical automated ML workflow in Azure ML starts with selecting the type of ML (e.g. classification or regression), point to the location of the training data, then decide where to perform the training (locally or various remote locations such as Azure Machine Learning Compute or Azure Databricks), and finally configure advanced features such as hyperparameters, number of iterations and preferred model evaluation metric. Different ML algorithms are then trained and presented in the order of performance.

In an MLOps approach (Machine Learning Operations, akin to DevOps), Azure also puts a strong focus on the lifecycle management of ML models through reproducible ML pipelines, streamlined deployment, model lifecycle tracking and monitoring. As SageMaker Azure Machine Learning is also a paid framework.

5.2.2.3 Commercial: DataRobot

DataRobot [179] is another ML lifecycle tool that provides ease of access and partial automation of many steps in the ML workflow, from data exploration to model selection and deployment. Once the data is uploaded and the target variable for prediction is selected (and optionally a feature selection is performed), DataRobot uses automated ML to train and evaluate a list of popular ML algorithm implementations. These algorithms are then ranked by accuracy and a recommendation based on accuracy and training speed is made to the user who can inspect a variety of aspects about the model performance. DataRobot is available in the cloud, on-premise, or as a fully managed AI service with flexible deployment capabilities.

5.2.2.4 Commercial: Valohai

The last commercial MLOps product in this list is Valohai, a workload processing system that offers machine orchestration, version control (e.g. for easy reproducibility of experiments) and pipeline management for machine learning [180]. It fetches the code, configuration and data sources and enables the pipelines to work on top of an automated infrastructure which can be controlled through a user interface. In similar fashion to other ML lifecycle platforms, Valohai governs statistical models through the whole process of data pre-processing, model training and deployment, but also covers termination for when the model reached the

end of its lifecycle. Valohai runs on top of all the major cloud-providers as well as custom on-premises hardware and supports automated ML, e.g. for data management, transformation and anonymisation, model training and management, and interactive deployment.

5.2.2.5 Open Source: Acumos

Acumos [181] is part of the Linux Foundation AI Foundation and aims to standardise the infrastructure stack and components required to run an out-of-the-box general AI environment. Acumos provides onboarding, management and deployment of ML models as Dockerized microservices. In the centre of it all is Acumos' federated marketplace that allows users to publish, rate, and collaborate on ML models publicly as well as privately. Acumos is also developing a visual user interface to build an ML pipeline without code, called Design Studio. Acumos is still in development as of October 2020. The Design Studio is in beta phase and the Marketplace contains only a handful of models. It has not gained traction yet.

What makes Acumos stand out, however, is that it claims to support SDN and ONAP as "*many Marketplace solutions originated in the ONAP SDN community and are configured to be directly deployed to SDC*" [182].

5.2.2.6 Open Source: H2O

H2O [183] is an open source alternative to big players such as Amazon AWS SageMaker and Microsoft Azure Machine Learning, providing an open source E2E ML platform that includes automated ML and easy deployment. One of the key differences is that H2O does not natively use Docker containers for packaging the ML models but instead uses a distributed Java Virtual Machine (JVM) setup. H2O also provides a paid version, called Driverless AI, which promises to "*speed up the data science workflow by automating data exploration, visualizations, feature engineering, model tuning, explanations, model deployment*" [184].

5.2.2.7 Open Source: Clipper

Clipper [185] is an open-source project by UC Berkeley. It provides a prediction serving system for ML models with focus on low latency. Clipper uses a Docker container environment for flexible deployment and can be used in an existing Kubernetes setup. Models can be deployed from within several ML software tools, such as Keras, PyTorch, TensorFlow or SciKit-Learn through the Clipper API or a REST interface. However, Clipper relies on third party ML tools like these as it is not an E2E ML platform. It does not cover the initial part of the ML workflow, such as data exploration, feature selection and model development.

5.2.2.8 Open Source: TensorFlow serving

A similar role in deploying ML models is TensorFlow Serving [186]. It primarily caters for deploying TensorFlow models but can be extended to serve non-TensorFlow models as well. Like Clipper, TensorFlow Serving covers the model deployment rather than the whole ML workflow. It is therefore not an E2E ML platform.

5.2.2.9 Open Source: MLflow

MLflow is an open source ML lifecycle management platform from Databricks. According to [187], it offers the following functionality:

- MLflow Tracking, to record and query experiments: code, data, config, and results
- MLflow Projects, to package data science code in a format to reproduce runs on any platform
- MLflow Models, to deploy machine learning models in diverse serving environments
- Model Registry, to store, annotate, discover, and manage models in a central repository

MLflow supports many third-party model types, including H2O, Keras, PyTorch, Scikit-Learn, TensorFlow and XGBoost, to save and load models. Models can be deployed to Microsoft Azure Machine Learning, Amazon AWS SageMaker and Apache Spark UDF.

5.2.2.10 Open Source: Kubeflow

Kubeflow [188] is an open-source machine learning platform with the focus on deployment of ML models on Kubernetes. It also provides TensorFlow model training capabilities as well as pipelines for an E2E oriented solution to ML lifecycle management and experimentation. Although Kubeflow is centred around TensorFlow, it also supports various other ML libraries, such as PyTorch, Apache MXNet and XGBoost.

Kubeflow can be deployed anywhere where Kubernetes is installed on premise but also in cloud on AWS, Azure, Google cloud, IBM cloud and RedHat OpenShift.

5.2.2.11 Open Source: Apache Spark

Apache Spark promotes itself as a "*unified analytics engine for large-scale data processing*" [189] and provides several libraries for an ML workflow, such as SQL, dataframes, data streaming, graph analytics and a machine learning library (MLlib [190]) with a wide collection of ML algorithms. MLlib is a scalable machine learning library that can be used from within Java, Scala, Python and R, and contains algorithms such as ML model building and evaluation, feature transformation and ML pipeline construction.

5.2.2.12 Open Source: OpenFaaS

OpenFaaS (OpenFaaS - Serverless Functions, Made Simple., n.d.) is an open source implementation of a Function-as-a-Service (FaaS) architecture. Serverless function serving brings together two closely related concepts: (1) serverless, a cloud computing execution model where the cloud provider provides the whole server architecture and the customer provides front-end application code to be executed (sometimes also referred to as Backend-as-a-Service); and (2) Function-as-a-service, where the focus is on the development of functions (atomic services that are smaller than microservices) while the server side is provided by the FaaS provider. These concepts allow scaling to zero in no-demand times and pricing (where applicable) is based on the actual amount of resources consumed by an application, rather than on pre-purchased units of capacity.

Besides commercial options such as AWS Lambda, Google Cloud Functions and Microsoft Azure Functions, OpenFaas offers FaaS functionality as open source. OpenFaaS can be set up in the cloud or locally, and can run on top of Kubernetes or Docker Swarm. When set up locally, and given the availability of suitable hardware, then OpenFaaS can be used with no costs and provides full control of the deployment.

While the concept of FaaS, and OpenFaaS in particular, was developed for generic function serving rather than specifically for ML model serving, it is well suited for ML model deployment as well. Here, OpenFaaS allows to easily deploy ML model code from any language, even pre-existing legacy code can be brought into OpenFaaS function format. Some languages are supported directly by OpenFaaS (e.g. Python, Node.js, Java, Go), but theoretically ML models of any language can be deployed because the ML models are wrapped into Docker containers. The use of Docker containers not only allows for language independent ML model deployment but also for flexible deployment and scaling as required by service-oriented architectures such as 5G-CLARITY.

5.2.2.13 Summary and preliminary evaluation

From the plethora of available ML platforms some are commercial while others are open source, some are trying to cover the whole E2E ML workflows including data exploration, while others focus more on the deployment of trained ML models. Some come with their own set of already implemented ML algorithms

that have to be used by the ML designer.

The 5G-CLARITY requirements defined in Section 5.1 rule out some of the described ML platforms for use in 5G-CLARITY because they are not open source, such as Amazon AWS SageMaker [177], Microsoft Azure Machine Learning [178], DataRobot [179] and Valohai [180]. Acumos [181] looks like a good candidate with its supposed synergy with ONAP but so far only its Marketplace appears to be released for use. H2O [183] is a strong candidate for an open source E2E ML platform, however, H2O is built on a Java Virtual Machine approach rather than Docker containers, which is in conflict with the envisioned containerised approach for the AI engine, including deployment of ML algorithms into the Docker-based RT RIC. TensorFlow serving [186] is centred around deployment of TensorFlow models. While it also aims to support other types, the obvious focus on the TensorFlow environment could be a limiting factor.

Other potential candidates for the 5G-CLARITY AI engine are MLflow [187], Clipper [185], Kubeflow [188], Spark [189] and OpenFaaS (OpenFaaS - Serverless Functions, Made Simple., n.d.). All are open source, but they vary in the features they offer. Among these, Clipper appears to be the least mature and Kubeflow requires Kubernetes, which may be conflicting with the requirement on low installation overhead. MLflow is focussed on Databricks and Python, and Spark does not natively use Docker containerisation. An initial investigation of the discussed software frameworks revealed that OpenFaaS may be considered as a basis framework for the AI engine as it supports all 5G-CLARITY requirements:

1. Open source (out of the box).
2. Minimal overhead for installation and long-term maintenance (out of the box).
3. Packaging of ML models as a format that enables seamless deployment into near real-time RIC component operating in the RAN cluster, which supports containerised xApps (out of the box or with minor modification).
4. Enable experimentation with cutting edge and custom ML algorithms beyond state-of-the-art algorithms (out of the box).
5. Allow for training of ML models to be inside or outside the private venue, e.g. in a public cloud with enough training resources (out of the box).
6. Enable the lifecycle management of ML models, including instantiation, removal, update of ML models (out of the box or with minor modification).
7. Provide a registry for ML models for ML service discovery (out of the box).

The final requirement will be fulfilled by integrating the Intent engine (Section 6) with the AI engine:

8. Enable access to ML functions to non-expert private network operators (after integration with intent engine)

5.3 5G-CLARITY initial design

The primary role of the AI engine is to act as a host for the ML models to be deployed, where some candidate models are described in Section 4. This includes offering ML algorithms as services to the network operator, keeping track of what ML services are available, and providing a way to manage their lifecycle. Figure 5.2 shows the AI engine in the context of the 5G-CLARITY environment, with connections to the network operator through a dashboard, to an ML designer through a model lifecycle interface, and to other 5G-CLARITY components, such as Slice Manager and Telemetry Collector that are part of the 5G-CLARITY management stratum and are described in detail in Section 2.

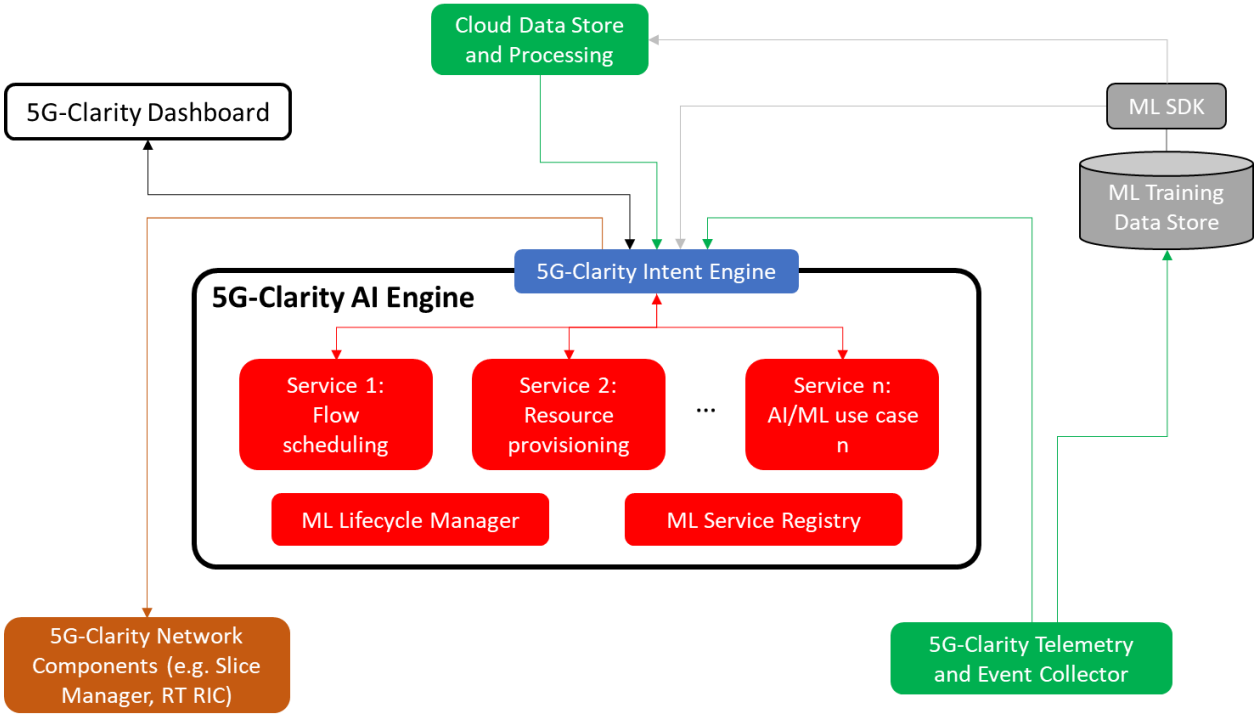


Figure 5.2 Architectural overview of the 5G-CLARITY AI engine.

A typical workflow of interactions with the AI engine consists of the following steps (visualised in Figure 5.3):

1. An **ML designer** trains an **ML model** (on or off premise).
2. The ML designer deploys the trained model in the AI engine, where the model becomes an **ML service**. The **ML Lifecycle Manager** supports the onboarding of the service and adds the ML service to the **ML Service Registry**.
3. Consumers of ML services, such as the **network operator** or other **5G-CLARITY network functions or application functions**, can execute any ML service that is available in the ML Service Registry.
4. ML services that are executed by a consumer may retrieve input data from the **5G-CLARITY Telemetry Collector** and may forward configuration suggestions to 5G-CLARITY network functions or application functions.
5. The ML designer may monitor the performance of the deployed ML services through the ML Lifecycle Manager and update or delete existing services upon demand.

A list of services that the AI engine offers in connection to the management and execution of ML services was introduced in 5G-CLARITY D2.2 [2] and is described in Table 5-1.

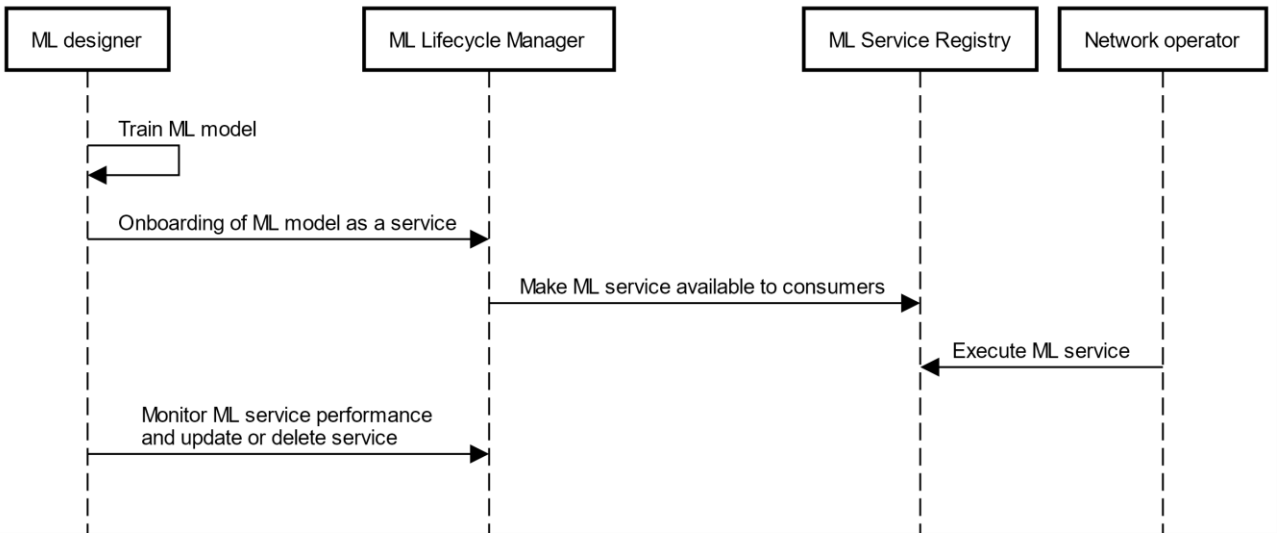


Figure 5.3 Sequence diagram of a typical workflow for management and execution of an ML service that is hosted by the AI engine.

Table 5-1 AI Engine Services.

MF Service ID	MF Service Name	Description
AIEngine_Ml_Mod_Mgmt	ML Model Management service	This service allows the deployment, update and removal of trained ML models by the ML designer. The ML model shall be provided in containerised form. It will consume input data from the Data Management and send output to various network functions (e.g. Slice Manager). Deployed ML models can be updated after they have been retrained, e.g. when additional data has become available. At the end of their life span, deployed ML models can be removed from the AI engine.
AIEngine_Ml_Mod_Register	ML Model Registry service	The deployed ML model can be registered in the ML Service Registry. In the case of pre-trained models, this service will allow the intent engine to discover ML services and connect them to a given intent.
AIEngine_Ml_Mod_List	ML Model List service	Lists the ML models that are currently deployed and ready for execution.
AIEngine_Ml_Mod_Run	ML Model Execution service	Runs a deployed ML model that is available in the ML Service Registry. Once executed, the deployed ML model may run as a once-off or recurrently.
AIEngine_Push_XApp	xApp Pushing service	Certain (real-time ready) ML models can be pushed down into the near-RT-RIC as xApps.

5.3.1 Containerised ML models

The basis for the offered ML functionality in 5G-CLARITY is containerisation of ML models using Docker containers. At a glance, the Docker containerisation environment enables developers and ML designers to wrap their code into isolated containers, which run inside a virtualised Docker environment on top of a host operating system as shown in Figure 5.4.

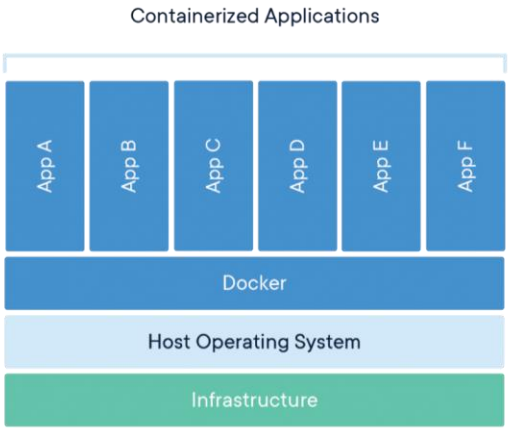


Figure 5.4 Conceptual layers of a docker environment, where multiple ML models run as containerised services[3].

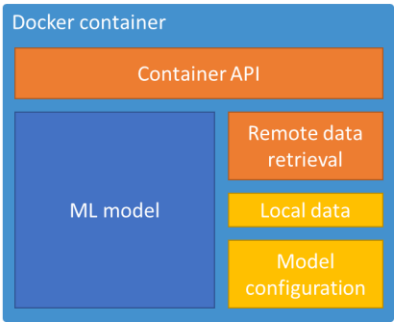


Figure 5.5 ML service packaged inside a Docker container, with the ML model (blue), data exchange (orange) and storage (yellow) components.

In the context of the 5G-CLARITY AI engine, each of the containerised applications from Figure 5.4 represents a packaged ML service. As shown in Figure 5.5, a 5G-CLARITY ML service docker container comprises the ML model and some utility components, mainly for communication and configuration.

5.3.1.1 ML model

The ML model is the core of the container. From the perspective of 5G-CLARITY, an ML model is essentially a function that takes a specific input and returns a prediction or recommendation output. In a typical ML use case in 5G-CLARITY, an ML model is a pre-trained model or a model that performs a continuous training inside the container. From an implementation point of view, the ML model can be an executable binary file or a piece of software code together with a compiler. The specific ML models that are planned to be deployed in 5G-CLARITY are described in Section 4 and include:

- Predicting SLA violations/success rate (Section 4.2)
- RT-RIC: AT3S traffic routing/handover (Section 4.3)
- RAN slicing in multi-tenant networks (Section 4.4)
- User association (Section 4.5)
- Data and Computation Offloading (Section 4.6)
- Indoor Ranging with nLoS Awareness (Section 4.7)
- Resource provisioning in a multi-technology RAN (Section 4.8)

[3] https://www.docker.com/resources/what-container

- Dynamic transport network setup and computing resources provisioning (Section 4.9)
- Adaptive AI-based defect detection in a smart factory (Section 4.10)

5.3.1.2 Container API

It is also essential that each container provides an API through which the ML model can be interacted with, as shown in Figure 5.6. The input to the model and the model's output will be transmitted through this API, and all models deployed in the 5G-CLARITY AI engine shall adhere to a common API specification. In this way, all ML models will communicate using the same protocol. For example: *i)* the model receives input data in JSON format; *ii)* it provides a "*predict()*" function that processes the input and returns a value or series of values as the prediction result; which *iii)* is returned in JSON format.

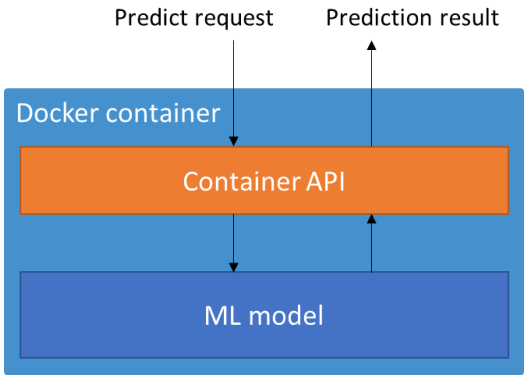


Figure 5.6 The Container API is the point of communication through which the ML model can be queried to produce predictions given the input data that is encapsulated in the prediction request.

The container API is typically implemented using REST (Representational State Transfer) or RPC (Remote Procedure Call). REST is a well-known protocol with well-defined semantics that is popular for web APIs, whereas RPC is typically used for server-to-server communication. Furthermore, the base abstraction of REST is resources. It is based on simple formats and transports (JSON/XML/HTTP), thus it is easy for system integration. However, JSON/HTTP is less efficient than other binary alternatives, such as RPC. RPC abstracts as function calls instead of resource objects. It is based on more complex formats (gRPC, Protobuffers, HTTP, Binary) and therefore requires significantly more effort to architect and integrate compared to REST. However, once RPC is implemented, it provides much better performance than REST due to optimised binary formats and transports. Both protocols are suitable for containerised applications and are used by existing ML deployment frameworks.

5.3.1.3 Remote data retrieval

Some ML models may require additional data to respond to a prediction request, for example the ML model may need additional context from the network or access to current or past telemetry. Therefore, the ML service container may contain a function to establish a connection to the 5G-CLARITY Telemetry subsystem, described in Section 2.4.2, to retrieve additional data as depicted in Figure 5.7. As described in Section 6 another possibility to facilitate the connection of the AI engine to the multiple data sources in the system is to have the use the Intent Engine as mediator.

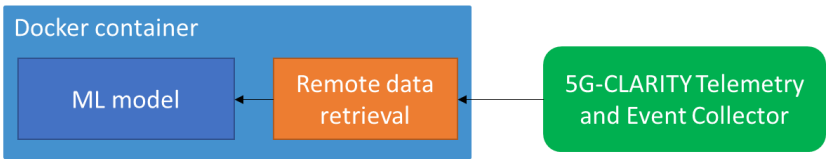


Figure 5.7 Some ML models may request additional network data from the 5G-CLARITY Telemetry and Event Collector.

5.3.1.4 Local data storage and model configuration storage

The containerised ML service may also include local data that help with a given prediction or network optimization task, such as frequently used telemetry data that is locally cached inside the container or model-specific data that was provided along with the ML model. A model may also store its hyperparameters or other configurations inside the container. This local storage of data and configuration is optional and may not be used by all ML models deployed in 5G-CLARITY.

5.3.2 ML service registry

The ML Service Registry maintains a catalogue of the available ML services, which are offered to the users of the AI engine. This catalogue shall be updated each time a new ML service is deployed or removed. ML models that are hosted by the AI engine but that are not ready for prediction queries shall not be listed as "available" in the ML Service Registry. This includes offline models that are currently training (such as Random Forest models) and online models in the early training phase that have not yet converged to a state of reliable predictions (such as reinforcement learning models). Models that have reached stable prediction state shall be added to the ML Service Registry.

5.3.3 ML model lifecycle management

An essential part of using ML models as services in the AI engine is ML model lifecycle management. This includes the onboarding, monitoring, updating and removal of ML models as depicted in Figure 5.8. These actions are typically taken by the ML model designer directly, who has the best understanding of the model and its performance.

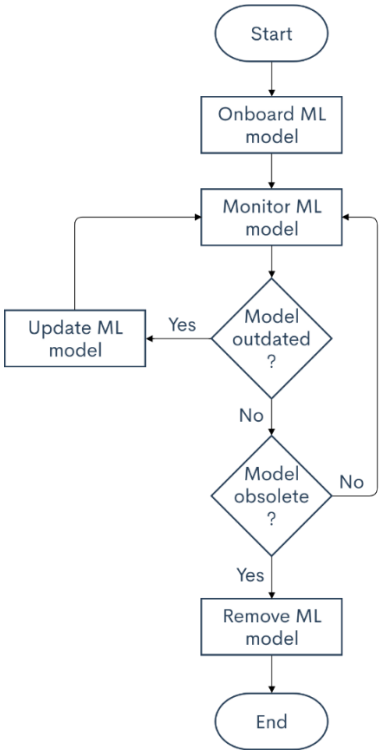


Figure 5.8 Typical workflow of ML model lifecycle management from the perspective of the ML model designer who is deploying a pre-trained ML model.

5.3.3.1 ML model onboarding as a service

The first step in the ML model lifecycle is the onboarding of the model into the AI engine, where it can either be deployed as a pre-trained model that is ready to return predictions, or be trained inside the 5G-CLARITY environment from within the AI engine, using live 5G-CLARITY telemetry data. An ML model that is deployed and available to deliver predictions (whether pre-trained or in online training mode) is defined as an ML service, which shall be added to the ML Service Registry where it can be discovered by users of the AI engine. ML models that are currently training and are not yet ready for making predictions shall be added to the ML Service Registry once they become available as fully trained ML services.

Since ML services are deployed in the AI engine as Docker containers, the ML model designer shall align the design of the model containers with forthcoming 5G-CLARITY guidelines to ensure compatibility with the AI engine. This includes, for example, an API for communication with the ML model and a method to retrieve 5G-CLARITY telemetry data if that is required by the model (see Section 5.3.1).

5.3.3.2 ML service monitoring

The AI engine shall provide methods to monitor the performance and general health of deployed ML models. The model performance relates to the prediction error, which can be obtained, for example, by comparing the values that were predicted in the past with the actual current values of a given variable or by monitoring the distribution of input data and prediction output over time to detect data drift. Other monitored metrics may include health-related information such as CPU and memory consumption. The monitored metrics may be visualised to the ML model designer through a dashboard such as Grafana as exemplified in Figure 5.9.

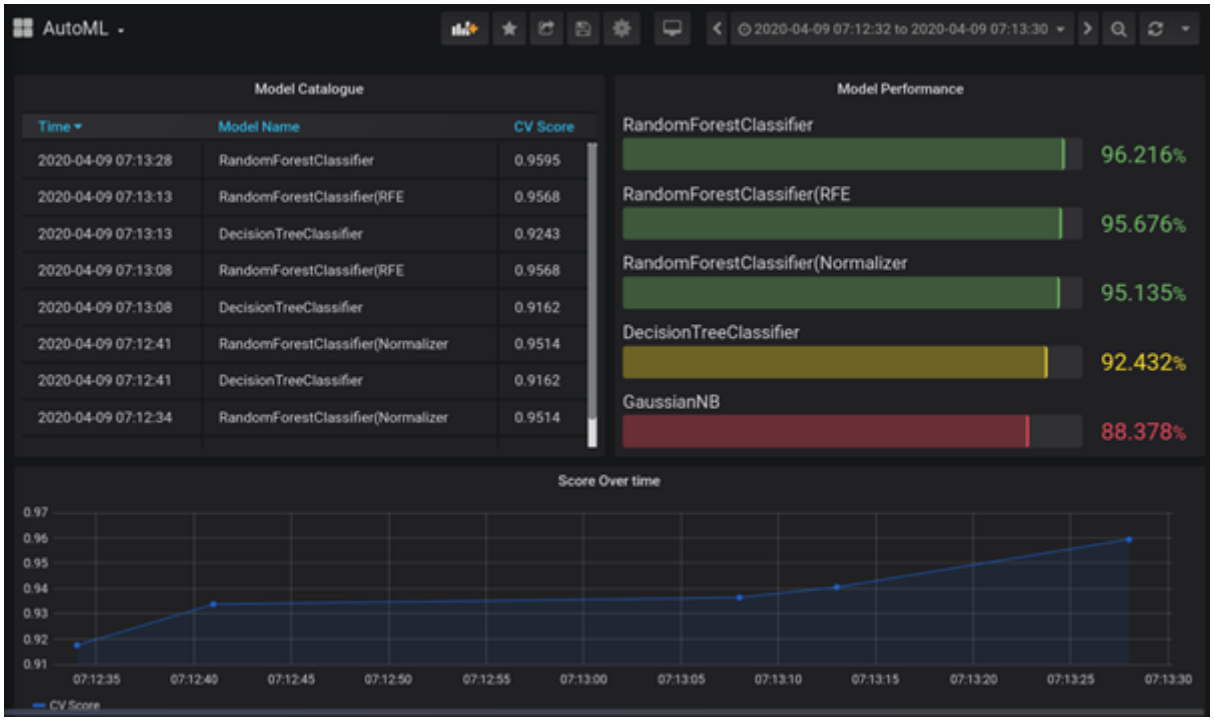


Figure 5.9 Example of visualisation of ML model performance [191].

5.3.3.3 ML service update

Another step in the lifecycle of a deployed ML service is the update of the underlying ML model. From time to time the model may need to be updated because of model decay, where the model loses accuracy over time due to a change in the data distribution (data drift) or change in the meaning of the data (concept drift). The ML model designer can detect a decaying model using the monitoring through visual inspection or by comparing the model performance to a pre-defined threshold. When a decaying model has been detected, it needs to be retrained and/or a new version of the model must be deployed.

The retraining of the ML model may take place within the AI engine or outside, depending on the capabilities of the deployed ML service container. Regardless of the process of re-training, a new version of the ML model replaces the old one in the AI engine, and the ML Service Registry must be updated accordingly.

5.3.3.4 ML service removal

The final step in the ML lifecycle workflow is the removal of ML services. An ML service may be temporarily removed as an alternative to updating, followed by a re-deployment at a later stage. In other cases, the ML service may be deemed deprecated because its functionality is no longer needed. Then, the ML service is permanently removed from the AI engine. In either case, the ML Service Registry must be kept updated with the status of the ML service in question.

5.3.4 Interfaces to and from the AI engine

As described in 5G-CLARITY D2.2 [2] the AI engine is located within the Intelligence Stratum of the 5G-CLARITY system, and therefore has interfaces through which it can communicate with other 5G-CLARITY elements. An overview of these interfaces is shown in Figure 5.2.

5.3.4.1 Network telemetry

Some ML Services require live network telemetry data, which they can retrieve through an interface that shall be provided by the network telemetry provider. Section 2.4.2 identifies the potential telemetry sources

within the 5G-CLARITY system that ML Services may be interested on. As part of our initial design, we consider that the access of the ML Services to the various telemetry sources is going to be mediated through the Intent Engine. Example use cases are provided in Section 6.

5.3.4.2 Cloud data storage and processing

Additional resources for storing and processing large amounts of data for ML model training can be outsourced to a (hybrid-)cloud solution in order to extend computation power beyond local resources if needed or to train on data that is stored in the cloud. Models that have been trained in a cloud environment should be onboarded into the 5G-CLARITY AI engine after the outsourced training has concluded through the ML lifecycle management onboarding process.

5.3.4.3 Intent engine (Intent interface)

The 5G-CLARITY Intent engine provides an Intent-oriented interface towards users of the AI engine. Some of the AI engine's functionality shall be accessible through the Intent interface to facilitate ease of access. For example, the Intent engine shall be the primary way to execute available ML services such that a non-expert network operator can run ML functionality through a simplified interface.

Other functions of the AI engine may also be accessible through the Intent interface, such as onboarding and removal of ML services. The Intent engine may also be the central point of access for the ML models to request telemetry data from the network and forward recommended configurations to the network. It will be investigated how much of the traffic to and from the AI engine can be routed through the Intent engine in practical scenarios, as invoking the Intent engine adds overhead to these requests.

More details and some use cases of access to 5G-CLARITY network functions through the Intent engine are discussed in Section 6.

6 Intent Based Networking

6.1 State of the art

The term and concept of Intent, namely the ONF Northbound Interface (NBI) group, was introduced to a wider audience during an Intent Summit in 2014. Subsequently, the ONF standardised an intent approach as their NBI (document TR-523, October 2016 [192]). The goal of such an intent-based interface was to further liberate (SDN) applications from details of the underlying network(s). Two SDx Central articles provide the state of discussion at the time: Marc Cohn summarises the summit [193] while David Lenrow details the intent model [194].

6.1.1 ONF northbound interface

Lenrow's article is important because it sets the scene for most of the intent-based networking artefacts we find today. He summarises an intent as "tell me what you want" and "don't tell me what to do", which is accompanied by a process in which an (intelligent) application "translates the intent into an infrastructure-specific prescription that causes the network to behave in the desired manner". Five main advantages (or characteristics) are identified as an Intent being (1) invariant, (2) portable, (3) composable, (4) scaling out (not up), and (5) providing context. The first four characteristics are easy to understand and important. The fifths characteristic though implies that the reason for the intent is known and thus we have a possibility to determine apparent conflicts and can find ways to mitigate them.

The before mentioned ONF standard TR-523 defines an intent as a declarative paradigm (or method) for interactions between service consumers and service providers via an intent-based NBI that takes "what" (the intent) from the consumer and forwards notifications from the provider. The Intent NBI properties are then a formalisation of the original intent characteristics as:

- Non-prescriptive: specify service request (what) and leave delivery and resource use to the provider (how).
- Provider-independent: same request (intent) can be presented to any provider, while terms from the intent are translated into the provider's world (using so called mapping lookups).
- Declarative: consumer declares, without request of further details.

A system realising the Intent NBI then makes use of continuous loop(s) comparing existing and new Intents, mappings, and controlled resources (or resource sets), and evolving states. The mappings used for the translation are realised using key-value stores. A "key" represents a consumer term and its value represents a provider term. When used accordingly, it allows to translate simple consumer terms to (more) complex and detailed provider terms.

A fully intent-based system (with consumer, NBI system, and provider) can then be used to manage the life cycle of networking infrastructure and/or desired network states. Four key characteristics are defined for such a system: (1) translation and validation process; (2) automated implementation using automation or orchestration; (3) awareness of network state; and (4) assurance (including dynamic optimisation and remediation) [195].

The ONF intent model has resulted in some open source projects. To give three examples, OpenStack Neutron used group-based policies as contracts between groups of end points [196], OpenDayLight developed network intent composition for connectivity intents (see for instance [197]), and ONOS built an intent framework [198].

6.1.2 IETF NMRG

The IETF NMRG has adopted the intent model as Intent-based Networking (IBN). This work has seen active development of draft standards for general concepts, learning and reasoning, classification, intent-driven network management, and intent-policies in autonomic networks (see for instance [199]). A number of these draft documents have expired in 2019. However, the concept of IBN is still a working item and a further drive in the IETF can be expected.

In the NMRG, an intent is defined as "A set of operational goals that a network should meet and outcomes that a network is supposed to deliver, defined in a declarative manner without specifying how to achieve or implement them."[200]. IBN then is a network that can be managed using intents. It also introduces a Single Source of Truth (SSoT), a functional block in an intent system that normalises intents and services as a single source of data (for lower layers). Using this definition, the IETF moves away from defining an intent as a type of policy. It also removes the dependency of intent on autonomic networks. An intent now provides a data abstraction as well as a functional abstraction.

An intent-based system is characterised by six properties: (1) single source of truth, (2) one touch but not one shot, (3) autonomy and supervision, (4) learning (not imperative such as ECA policies), and capability exposure (for intent composition), and (6) abstraction. While the general process of an intent is still very close to the ONF origin, learning and supervision are important new concepts here. Furthermore, we can find intent categories as terms for further discussions: operational, rule, service, and flow.

6.1.3 ETSI and 3GPP

ETSI and 3GPP have seen an increasing interest in intent-based approaches, for instance in the zero touch and network intelligence initiatives (ETSI ZSM, ENI) and SA5 architecture (3GPP). These work items seem to be closely related to network automation and autonomous networks (which are important if not necessary to realise intents as the ONF has stated). Furthermore, the TM Forum ZOOM project has started a work stream on intent-based resource management aligned with ETSI MANO (see for instance [201]).

In 3GPP, TR28.812 defines an intent as "A desire to reach a certain state/position for a specific entity for instance for a service assurance or network deployment task" [202]. Notably, the intent does not define the necessary steps to get to the wanted state. An intent-driven management service then is a "a management service that allows its consumer to express an Intent". The service connects consumers and producers. A consumer states an intent and a producer realises it by performing network management tasks or formulating and activating network management policies. Here we find a clear distinction between intent and policy in the area of network management. The document then develops a layered model for intent-based interactions, intent expressions, and shows the dimensions of an intent-based framework. 17 scenarios for intent-based radio systems are presented.

6.1.4 Research

Calegatti et. al [203] provides an overview of research publications categorised as single or multi-domain solutions. The term domain is used here to describe technologically different parts of a network, for instance radio access, wireless access, backhaul, core, cloud, or SDN. In all single domain papers, an intent is a representation of a policy or a set of policies, which are activated or triggered once the intent is activated (or deployed, most papers do not use life cycle stages for intents). In these papers, we do not find examples of formal intent expressions.

Davoli et. al [204] describes a full intent framework, including intent expressions, for three different domains: IoT, data center, and transport network. Each domain is evaluated for feasibility, demonstrating that intent-based networking can be a valid even under near-real-time constraints. Esposit et. al [205] uses behaviour-

driven techniques to allow for intent expressions that are close to natural language (in this paper English using Gherkin). This approach is evaluated in three different scenarios. Arezoumand et. al [206] uses graphs to express connectivity between networked components, which are then used as intents. This approach requires only two network policies: forward and block. The mapping function is then the only variant part of the intent system.

6.1.5 Future work items

Calegatti et. al [203] also discuss the open issues for multi-domain intent-based systems. Even with standardisation organisations driving the topic, we still do not find much of language specifications or formalisms; at least not outside single-domain or proprietary use cases. The main question here is what semantics does an intent cover and how they can or should be expressed, then how they can be efficiently encoded in some form of language.

Subsequently, the two main functions for intent processing using such formal expressions are also far from being well understood: translation and validation. The translation of an intent depends on the expressiveness of the intent language, the involved semantic concepts, and the available processing power. Requirements are often driven by the application scenario, e.g. real-time translation inside an OSS, near-RT translation for an analysis inside a BSS. The validation of intents turns out to be as difficult as the validation of policies. Intents (despite the original claim) have not shown to ease the automate conflict identification and resolution; data consistency is still as problematic in intents as anywhere else.

6.2 Intent domain model

To understand the potentials and power of intent, beyond the discussed state of the art, we need to first investigate its applicability within system and application design. A fundamental principle here is the separation of mechanism and policy. This separation leads organically to where intents can be placed, what role in a system they can play, and major properties an intent has. The result is an Intent Domain Model (IDM), which describes all aspects of intent within its own field, before we will apply it to build an intent engine and describe how 5G-CLARITY will use intents.

6.2.1 System, mechanism, policy, intent

The separation of mechanism and policy is a fundamental design principle for systems. Here, a system is anything built for or with a purpose, e.g. an application, a product, a service, a platform, or a solution. Each system then has invariant parts – its mechanisms – and variant parts – its policies. Mechanisms of a system do not change or hardly ever change (over the lifetime of the system) thus they are invariant. To adapt mechanisms to required behaviour, they provide some means of governance, i.e. some interface which can be used to change their behaviour.

This interface is not exposed outside the system. Instead, the system then has one or more policies. These policies are variant, which means they are the variant part of the system (thus its mechanisms). Variant here means relatively dynamic, can change frequently over the lifetime of the system and much faster than any mechanism. To apply policy control to a system, the policies must use the offered means of governance of each mechanism. What a policy is depends then on (a) the concrete system and its mechanisms; (b) how those mechanisms chose to be governed (if they cannot be governed, then there are no policies!); and (c) how those mechanisms can be governed (if they cannot be governed then policies might exist but policy control is impossible).

Both, mechanisms and policies, can be relative. In a multi-layer or larger system for instance, lower layer (or level) policies can be higher layer (or level) mechanisms. This allows for hierarchical, recursive, layered, or fractal system design and composition.

When designing a system, one should always aim to maximise invariants (mechanisms) and minimise variants (policies). In software design this should lead to larger reusability, either of code through libraries or inheritance or as a runtime property because a small amount of assets provide principle services (invariant) rather than a huge amount of assets providing specialised (invariant) services. Once all mechanisms have been identified, the remaining variants are very likely policies. As a further note: in system architectures and models, invariance is the targeted abstraction, while variance is the option of multiple different implementations of such an architecture.

To give an example, take the heating system in a family house. All pipes, the tank (gas, oil, or kerosene), the heating device, radiators, and a water boiler are all mechanisms of the system. The policies are then the ability to switch the system on/off, to change the valve on the radiators (thus changing the amount of heat generated when switched on). Some of these policies can also be controlled via any electro-mechanic or electronic control unit automating on/off periods. The way these policies are offered to a user are then the actual valves on the radiators, the on/off switch, and the interface of the control unit. Their functionality is determined by the available policies, while their design and look & feel can be realised in any way, shape, or form desired, from simple switches, up to wireless control or even remote control with a smartphone application.

These interfaces to a system's policies is where intents are being used. While traditional interfaces might be based on exposing some process functions (e.g. flexible programming of on/off timers in our heating system), intents can be used to abstract the policy interface focusing on what a user (or another system) can achieve (govern) without the need to detail workflows or processes or actual programs. In other words, **an intent is an abstraction of a policy interface, to access (and change) a policy, which in turn govern a system's behaviour**.

Applying this view of intents to an Operation Support System leads to deeper understanding. Figure 6.1 shows nine different scenarios for managing a RAN, or parts of it, using intents. The examples start with an eNB (example a and b) and become more and more complex until an intent is used to manage a compete OSS cluster (example i).

In a first iteration of abstracting policy interfaces to intents, we may identify three different types of intents: managing single elements (a, b), managing OSS instances or cluster (c, i), or managing SON functions and SON coordinators (d through h).

These abstractions are based on the purpose the intents are created for, which ultimately determines the scope in which those intents may operate. Simple management of elements will focus on element functions, including initialisation, configuration updates, and profiles. SON functions and their coordination extends the scope towards finetuning the self-organisation of cells or tracking areas or other cell groupings. Finally, the management of an OSS or an OSS cluster will include all realised OSS functionality, e.g. assurance.

The scope of the intent could also be defined in different terms, covering all shown scenarios. For instance, the intent might be defined to manage one or more UEs (this is what O-RAN facilitates with its A1 interface for the none-real-time RAN intelligent controller, see for instance [207]). In this case, the scope of the intent is not directly related to the system, thus we will be able to use the same intent (i.e. only one intent type) on eNBs, SON functions, and OSS instances.

In other words, when we change the abstraction and scope of an intent from system towards function (or functionality), then an intent becomes inherently reusable across a wide range of very different systems. We can reuse existing policy interfaces and design a (compared to the systems, mechanisms, and policies) rather simple façade towards intents that abstract them into functionality.

Looking further into the façade, another important aspect becomes visible. An intent does not only abstract a policy interface but can also harmonise otherwise disjunct models and vocabularies. Figure 6.2 shows how

the façade works. An intent is declared somewhere, say in domain A using vocabulary V, then translated into policies or actions somewhere else, say in domain B using vocabulary W. The intent and the translation process harmonise two domains A and B (potentially including domain models) and different vocabularies V and W (with syntactic and/or semantic translation between them). The process is then reversed when policies and actions from domain B report on the intent fulfilment back to domain A. The figure shows a few example domains: network and element management (ENM, the Ericsson Network Manager product), databases, security, RAN, or local networks.

What an intent can express as well as how this can be translated depends on the policy interfaces of the systems. Figure 6.3 shows a closed control loop with analytics, decision making (policy), and orchestration. This control loop is a generalisation of ONAP and other loops used in the telecommunication industry.

Taking ONAP as the example, policies here are defined in XACML, Drools, or programmatic using the APEX policy engine [208]. The interfaces to these policies are standardised in the ONAP Policy Framework (PF) using TOSCA templates. The policies are then deployed as part of a service, represented in the figure as a closed control loop. Looking at the TOSCA templates allows to understand the scope of potential intents. Taking a defined (and then also deployed) service, allows to understand the potential intent instances for this particular service. This means we can build intent infrastructure, its core being the façade introduced above, for ONAP services in general, and then intent instances for any ONAP service in particular.

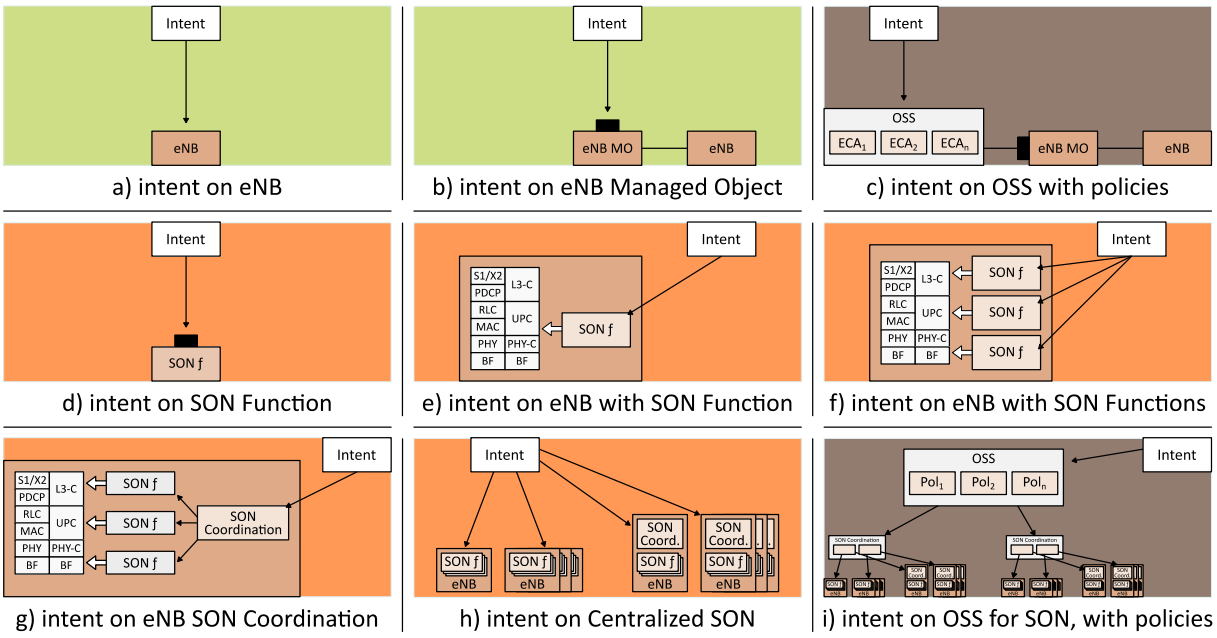


Figure 6.1: OSS intent examples: manage radio features with 3 intents.

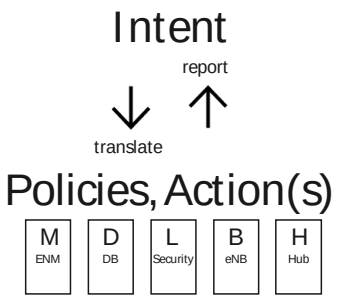


Figure 6.2: Intent, actions, and process.

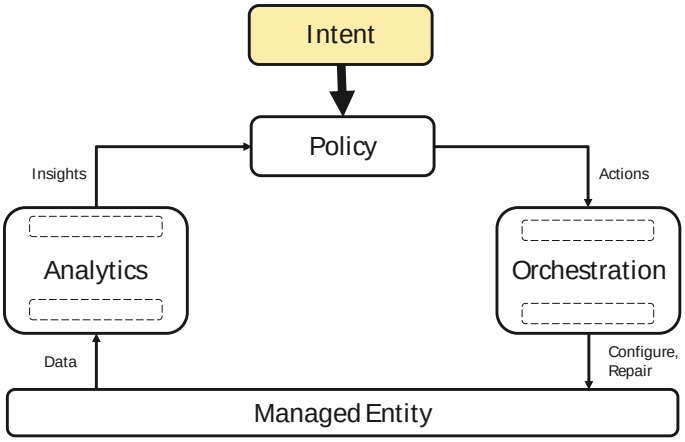


Figure 6.3: Intent and policies on a control loop system.

Careful design of the intent specification and the intent infrastructure will lead to virtually universal applicability, without rebuilding software or runtime assets. Careful selection of intent scope (as discussed above) with focus on functions (or functionality) rather than networks or elements, will lead to virtually universal intents that can be used over a wide range of networks and services. These statements might not (will not) hold if intent is used randomly in a system.

6.2.2 The essence of intent

Intent is often reduced to be the *what* rather than the *how*, mainly to state that an intent is declarative and not imperative. However, this reduction becomes problematic very fast, thus is not sufficient to define an intent. Take some network related intent examples:

- AI wants to improve quality using orchestration
- OSS wants to upgrade network to MBB
- Control loop wants to create 5G NFV chain
- Control loop wants to optimise application coverage-
- NOC wants to run AI exploration on specific network performance data

All examples focus on what is wanted. However, in reality these intents will carry more information, e.g. when is it wanted, how is it wanted (in what form, with what extended properties), when (or for how long) is it wanted, maybe even why it is wanted. We can look at systems using intents to see how many what/when/where/why/who/how properties do exist (SQL queries, every CLI interface, CM systems, markup languages, navigation systems, virtual assistants).

The actual important element of an intent definition are not the question words but the verbs being used. An intent defines “who wants what, potentially also when, where, why, and how as in what form”. This is then translated into policies and actions determining “who does what, when, where, why, and how” to realise an intent. As discussed above, this translation might cross domain boundaries. This definition of what does an intent express, is shown in Figure 6.4.

The figure also shows the communication between intents and actions (or policies). The process is refined to reflect where intents are being used. In network management, an intent might describe a required configuration. Intent fulfilment then depends on this configuration being maintained until the intent terminates (or is terminated). A continuous loop is required for intent assurance. This loop takes the intent, translates it into actions (at the time the loop executes), monitors the system, validates system behaviour, and reports the results. If the intent is not terminated, then the loop continues, starting with a new

translation, which will incorporate the current situation and might result in very different actions being selected. Since the translation is not part of the intent assurance, we use mapping (to map) to describe the translation of an intent towards a policy interface including domain boundaries.

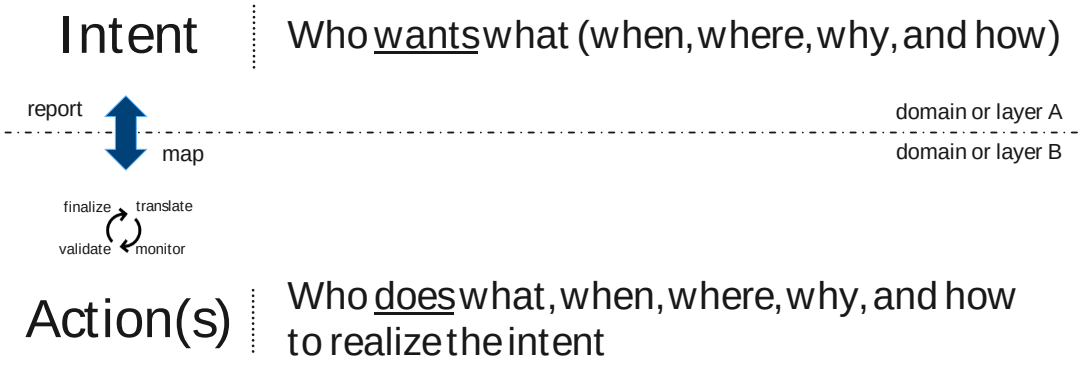


Figure 6.4: Intent in a nutshell.

To explore intent properties, we introduce four intents described in iconographical form (see Figure 6.5). The first intent states “get a taxi to a location”. Location might be further specified as “home”. The second example states “get a taxi to a sport event, at a specific date and time, using the lowest possible cost”. The third example states “get a taxi to a sports event, be environmentally friendly, at a specified time, using a specified payment method”. The fourth and final example states “get a taxi to a location, with a route to meet or collect a person, as a recurring event”. The term “taxi” might be generalised to “transport” or “lift”, which then would include not just regular taxi services but also chauffeurs, specialised transport services, or any person legally permitted to realise the transport.

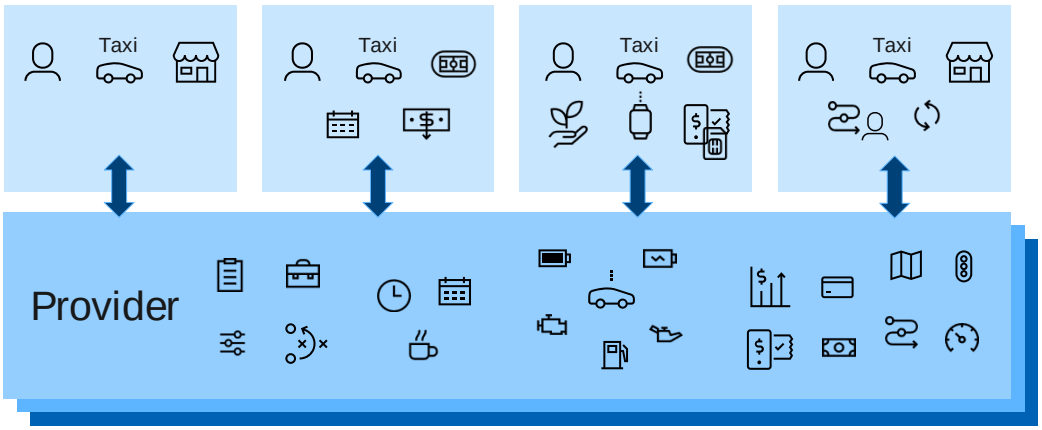


Figure 6.5: Iconographic intent examples to get a taxi (or transport).

The first observation, as shown in Figure 6.5, is that many different concrete service providers might be able to fulfil all requirements. This would be either different taxi companies, or other entities as mentioned above. Each of these providers maintains their own business model, including how they organise cars (including maintenance and legal requirements), schedules (of drivers and cars), cashflow, profit margins, routes, and whatever else their operation requires. None of that needs to be exposed, these are essentially all mechanisms. What is exposed is some interface to a person to book a taxi (or transport). This interface is represented by the dark blue arrows in the figure.

6.2.2.1 Primary objective and secondary objectives

Intents have one (1!) primary objective and may have 0 or more secondary objectives.

The primary objective of the four examples is “get taxi” or more generalised “get transport”. The primary objective can then be expressed as an action sentence, in English in the form of subject, verb (action), and object. “Who wants what” becomes now “Jill wants a taxi” or “Joe wants a chauffeur”. This primary objective is none-negotiable, whatever is described here is the basic requirement. The better the scope for the primary objective is set, the more reusable an intent becomes. For instance, “get taxi” excludes chauffeurs, while “get transport” includes any (legal, insured) transport service.

Secondary objectives may be used to further specify the intent in terms of what is exactly wanted, how is it wanted (in what form), when is it wanted, why is it wanted. Allowing a further specification on what is wanted makes the intent “get transport” useable in specific scenarios as “get transport, a taxi”.

Providers might require secondary objectives, even in defined syntax and semantics. Taxi companies for instance often require stating the destination. “go home” might be enough (if translated to the person’s home address). “N37 PV44” might be acceptable if everyone understand that this is an Irish zip code that points to an exact location (house).

The taxi examples also show a few more secondary objectives: at a future date (for the scheduled sports event), with cost constraints (lowest possible price), set taxi type (electric car), set payment method (via phone), with time constraints (at specified time, as a recurrent event), and with way point on the route (to meet or collect someone). These secondary objectives might be negotiable and may be combined using logic (I want “this and that” or “this or that”).

6.2.2.2 Temporal aspects

Intents exist once issued (or deployed) until fulfilled or terminated.

Intents can require instantaneous or scheduled delivery, as one-off or recurrent event.

An intent that is not issued or deployed cannot be considered by a system, it might simply be a template or potential intent kept in a repository for future use. Once issued, an intent exists in a system. The end of an intent is reached either when it is fulfilled (varying options to determine this state) or actively terminated (ideally by the issuer). The fulfilment of an intent depends on its requirements on delivery.

An intent can require instantaneous or scheduled delivery. Instantaneous means “now”. In our taxi examples at the time someone contacts a taxi company. Scheduled means at some time in the future, e.g. when booking a taxi for a particular time. These two types of delivery have implications on the intent provider, including risks (e.g. scheduled intents might be revoked, cancelled, voiding any effort already taken). When no time constraint is set in the intent, an instantaneous delivery should be assumed.

Furthermore, intents might be a one-off (realise, deliver, finish) or recurrent (delivery at specific times or continuously). Recurrent intents might have an end date (final fulfilment can be determined). If no end date is given, a continuous delivery until termination should be assumed.

Temporal aspects of intents require an understanding of time (and date). Figure 6.6 shows important concepts. The top shows common time properties, encoding formats, units, windows, date/time combinations. These are important in human-machine communication scenarios, and the human understanding of time expressed in an intent needs to be translated into a system view. This translation (and required calculations) is shown in the middle row of the figure. The concepts are instant, point, window, and period, while calculations can be done using the Allen’s interval algebra [209]. Each specification of time needs to be bound by a time axis and scale, which determines what an instant (or point) is and how small windows can be. Furthermore, time information needs to be provided in context (e.g. time of cause, effect,

monitor, normalisation, process, etc). A common, formal, ideally standardised understanding of temporal aspects and time is essential for any intent process.

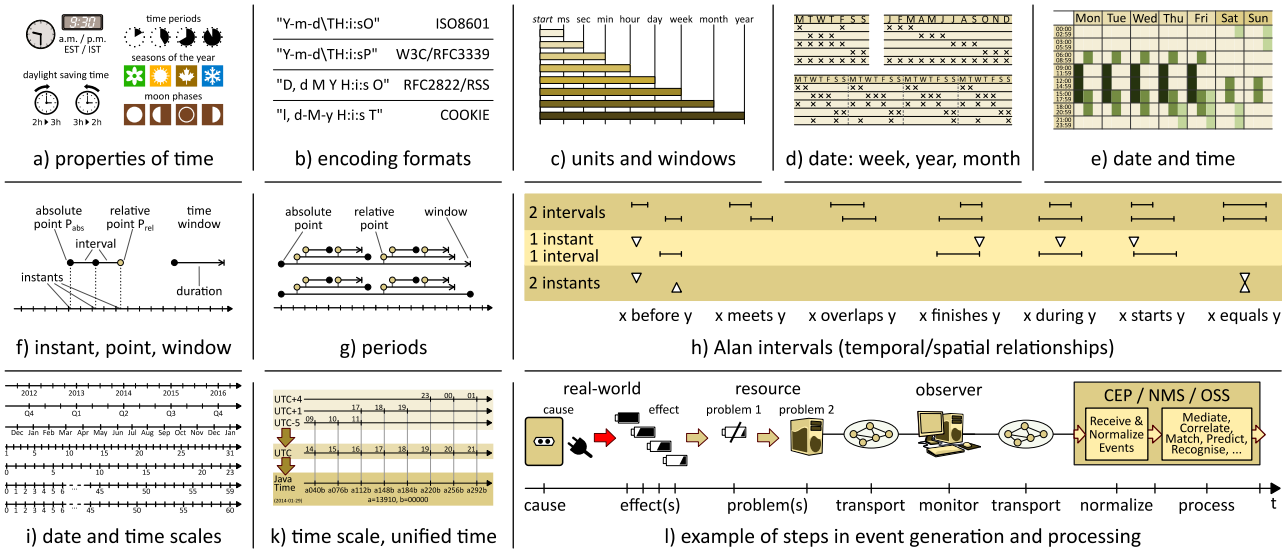


Figure 6.6: Time and temporal aspects (general).

6.2.2.3 Intent scenarios

Many intents, many providers, many domains, recursive.

There are four essential scenarios for intents, as shown in Figure 6.7. First, a provider can support many intents. While most systems will be designed for a single, specific purpose (and thus provide only intents for this purpose), some systems might aggregate a larger number of functions. An OSS for instance will support most of the FCAPS, thus provide intents for fault, configuration, accounting, performance, and security management.

Next, a carefully designed intent (primary objective) can be supported by many providers. The example intent “get taxi” can be provided by any taxi company. The more general intent “get transport” can be supported by any transport organisation or individual. Important is that the design of the intent will determine which type of provider can support it.

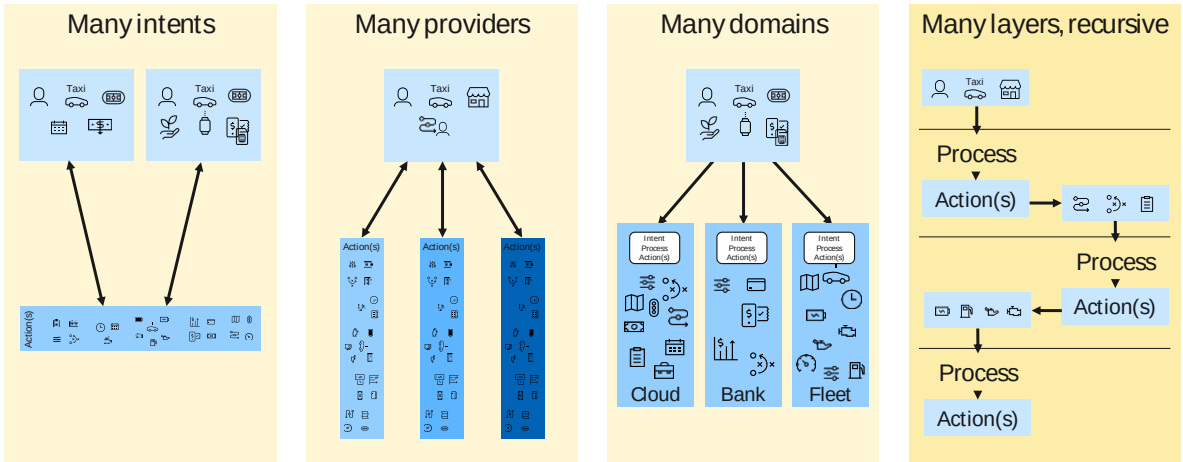


Figure 6.7: Intent Scenarios.

An intent can also be supported in many different domains, by one or more providers in each domain.

Mapping might be required to cross domain boundaries. Finally, intents can be used between many layers, in a recursive way (i.e. the number of layers is not determined a priori). In this scenario, the selected actions from one intent process are intents used as input for another intent process. This can be done recursively, until one or more actions are selected that are not intents anymore.

6.2.2.4 Declarative by design

An intent is declarative by design, i.e. not by definition.

This is a consequence of an intent being the abstraction of a policy interface, the separation of mechanism and policy in a system, and the maximisation of invariance in system design. The later results in an invariant system process (or control flow), which can be governed by policies. For instance, a protocol handler will create a PDU including a checksum. This is its mechanism, including the required control flow. The way the checksum is calculated can be a policy, e.g. using SHA-1 or MD5 or CRC algorithms. An intent is an abstraction of the policy interface. It can never interfere with the mechanism or control flow. Thus, it is by design declarative.

A declarative intent may contain logic in its specification. Relationships between the secondary objectives can be expressed using Boolean logic or even algorithms. The intent itself is still declarative. This is similar to declarative programming.

6.2.2.5 Processing intents

Intents can be processed in three phases, invariant.

Processing intents can be realised as a mechanism, an invariant process, with three phases. Figure 6.8 shows the phases with tasks (left) and what the tasks are doing (right). Phase 1, called initial, gathers an intent and maps it to an interpretable expression, potentially crossing domain boundaries. Some interface is required to gather the intent. This interface can be graphical, command-line oriented, an API, or any other interface that is intelligible for the intent requestor (a person or machine). Once the intent is gathered at the interface, it can be mapped from the requestor's domain into a domain with intent providers and translated from the interface into primary and secondary objectives.

The second phase is the continuous process of translate, monitor, validate, and report. For as long as the intent is deployed (see temporal aspects above), it is translated into actions. In the simplest form this is a direct selection of actions. In more sophisticated processes, machine learning and other techniques might be used.

Multi-stage process with complex tasks

1. Initial
   - Gather / collect intent
   - Map to interpretable expression
2. Continuous
   - Translate
   - Monitor
   - Validate
   - Report
3. Final (optional)
   - Report, summarize

What do the tasks actually do?

1. Interface (UX, API)
   - Requestor- / domain-specific expression(s)
   - 5W1H with domain model and context
2. Learn, automate
   - Select actions with required effect
   - Monitor the effect(s)
   - Check wanted versus current situation(s)
   - Report activities, outcome, projection
3. Finalize
   - Provide feedback or request more details

Figure 6.8: Intent process.

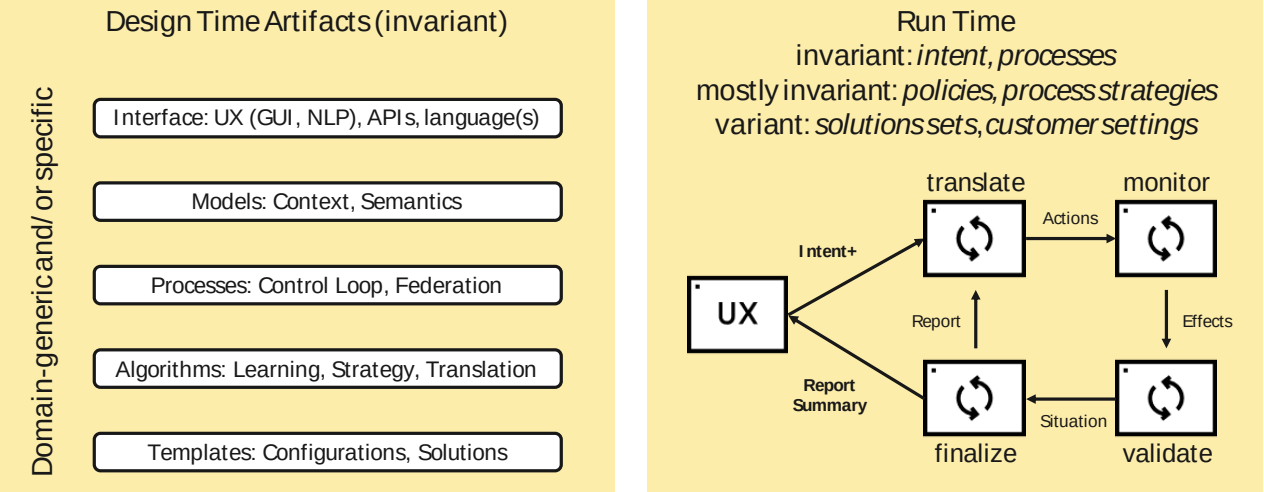


Figure 6.9: Intent process artefacts.

Once translated, the actions are executed, realised, enforced, or delegated. Then monitoring will collect information about the results. This information can then be validated, i.e. the actual effect of the actions being compared to the calculated and originally required effects. A report finalises one iteration of this loop. Reports might trigger a new iteration, in case repair actions are required. The last phase finalises the intent. This will happen automatically if the intent is fulfilled. Otherwise, it is triggered by an intent cancelation.

This general process can be realised to fit requirements of any intent provider. At design time (see Figure 6.9, left side), all required artefacts are invariant: interface (intent gathering), models (domain vocabulary), processes (control loops for continuous phase, federation for crossing domain boundaries), algorithms (for each of the process tasks), and templates (for intents, configurations, or solutions).

At runtime (see Figure 6.9, right side), the intents are (largely) invariant. So are the processes and tasks. Policies and process strategies should be mostly invariant, though each task might offer interfaces to govern its behaviour (these tasks are small systems and separation of mechanism and policy applies to them as well). Building products, services, or solutions can be supported by customisation (of tasks, the interfaces, the whole process) or solution sets (pre-defined settings and tools for the intent process).

6.2.2.6 Conflict identification and mitigation

Intents facilitate conflict identification and mitigation.

In classic software systems, access to mechanisms is realised directly or via direct access to the system's policies. In simple systems and with simple policies (e.g. parameters, simple functions), conflicts can be largely avoided. In a heating system for a house for instance, conflicts do not occur. As soon as the number of policies or their complexity increases, conflicts are becoming possible. Concurrent access to system policies make conflicts almost unavoidable.

Using intents does not eliminate potential conflicts. However, it facilitates their identification and mitigation. This is a consequence of explicitly declaring primary and secondary objectives (not actions) and translating them, continuously, into actions and policies. Providers that handle multiple intents simultaneously can identify conflicts on the primary objective (an overlap of contradicting objectives in a given special-temporal context). Evaluating secondary objectives allows to identify conflicting resource use, potentially in advance. Since actions (or policies) are selected continuously (in each iteration of the continuous intent loop), runtime conflicts can be identified and avoided in the translation process.

6.2.3 Application and resource model

An intent engine and its infrastructure can be built as an application (single, local execution of all tasks) or a distributed application (distributed execution of each task or task groups). 5G-CLARITY focuses on applications for and in networks, thus the intent engine is a distributed application. A single application model defines a blueprint for the implementation, more than one if required. It also serves as an abstraction of applications and resources that intent providers manage in order to fulfil network intents.

Figure 6.10 shows the UML class diagram of an application and a distributed application. An application is a program in form of a file that can be executed on processing systems as an application process. It has units, which are executed as application tasks. An application process can be member of a facility, which defines its scope. It can also be a member of a domain, which defines a common range for itself and all its tasks. A computing system is the collection of all processing systems under the same management domain with no restrictions of their connectivity. Tasks manage some resources.

This model covers simple and distributed applications, with units (of any width or depth) for the application logic. Some infrastructure is required, similar to operating systems: memory, input/output, and task management; distributed applications will also need shared memory management. Three orthogonal groups capture all application aspects: execution (system), scope (facility), and management (domain).

In a network, every application, their parts, and resources can be modelled as executables. Things that are not executed cannot be accessed (they are some assets in some repository or otherwise outside the network). Things that cannot be executed do not exist for any network application. Modelling all of them in a similar way allows for a unified application and resource model, supporting system design and intent processing.

Figure 6.11 shows the UML class diagram for executables. The core of the model is a unit, modelled using the composite pattern. This captures everything from large monoliths up to micro-segmented applications. Each unit has interfaces (input, output, management), parameters (simple policies), decision policies (e.g. with logic), and strategies (policies as logic). Units are mechanisms, policies are policies.

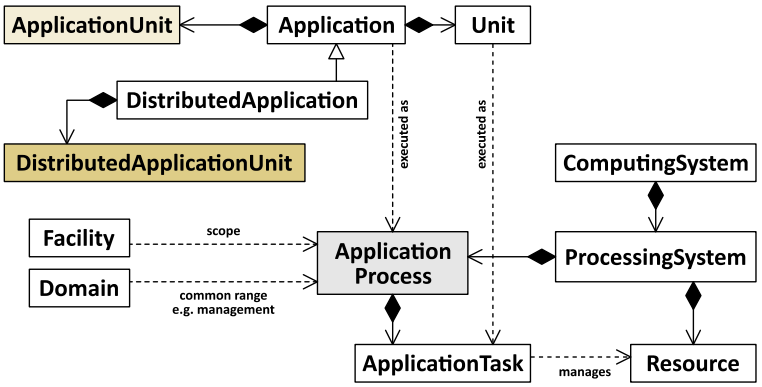


Figure 6.10: Application model (UML).

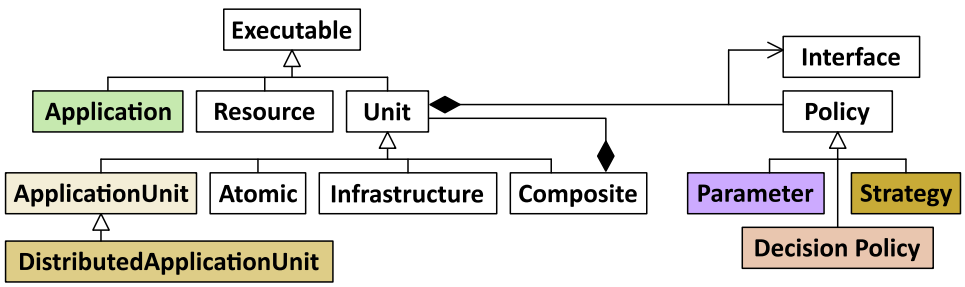


Figure 6.11: Application model, executable (UML).

This model can be expressed formally as follows. Let A be a set of application and an application a_j the tuple $\langle n, d, \pi, U^i, U^a \rangle$. Each application has a name n (also known as identifier), a description d (human readable text), and a type π. The type can relate to application classes, to categorise applications in catalogues, e.g. SDN applications as π^{SDN} and NFV applications as π^{NFV}. Each application requires some infrastructure U^i and has some units U^a.

$$A = \{a_{1..n}\}, a_j = \langle n, d, \pi, U^i, U^a \rangle$$

Let A^d be the set of distributed applications and a distributed application a_j^d the tuple $\langle n, d, \pi, U^i, U^d \rangle$. The only difference to an application is that the units are now distributed units.

$$A^d = \{a_{1..n}^d\}, a_j^d = \langle n, d, \pi, U^i, U^d \rangle$$

Let U be a set of units then a unit u_j is the tuple $\langle n, d, U, I, F^u \cup \Xi^u \cup P \rangle$. Units have a name and description. They can also have subunits U. Each unit exposes a set of interfaces I. And it can have policies in form of parameters F^u , strategies Ξ^u, and decision policies P.

$$U = \{u_{1..n}\}, u_j = \langle n, d, U, I, F^u \cup \Xi^u \cup P \rangle$$

An application process a_j^p essentially executes (has) a set of application tasks A^t.

$$A^p = \{a_{1..n}^p\}, a_j^p = \langle n, d, A^t \rangle$$

An application task is an executed application unit. Let A^t be a set of application tasks (of an application process) and an application task a_j^t the tuple $\langle n, d, \pi, A^t, I, F^t \cup \Xi^t \cup P \rangle$ (name, description, and type as standard). It may contain other tasks (A^t, like a unit can contain other units), exposes interfaces, and has policies.

$$A^t = \{a_{1..n}^t\}, a_j^t = \langle n, d, \pi, A^t, I, F^t \cup \Xi^t \cup P \rangle$$

An interface provides access to a unit (or task). It has a name and description as standard, plus some expressions λ. Fundamental expressions in CURD are create, update, read, delete (as defined in REST). In network management, we use create, delete, read, write, start, stop, and cancel-read (stop notifications). These are all operations ever needed on objects. All other operations are combinations and variants of them. These expressions can be accessed in different ways: API (function, command), RPC (function, command), pipe (event, stream, command), stream (event, command), file (event, command), REST, and others.

$$I = \{i_{1..n}\}, i_j = \langle n, d, \lambda \rangle$$

It can be useful to further distinguish types of interfaces, for instance operational, streaming, and management interfaces. An operation interface would provide one or more computational operations along with possible response to invocations. Streaming interfaces would be the endpoint of a stream as source or sink, potentially aggregating more than one flow (audio, video, etc.). Management interfaces are operational interfaces whose scope is management.

A set of resources R is part of the infrastructure of a processing system (e.g. storage, I/O). It can also be part of an application as application objects, e.g. MIB or RIB or even CMDB. A resource r_j has a name and a description along with its interfaces and a set of defined values F^r.

$$R = \{r_{1..n}\}, r_j = \langle n, d, I, F^r \rangle$$

A processing system is hardware and/or software capable of executing applications. It has a name, description, can be located (in some location model), has hardware and software resources, and zero or more executed application processes.

$$\Psi^p = \{\psi_{1..n}^p\}, \psi_j^p = \langle n, d, \check{l}, R^{hw}, R^{sw}, A^p \rangle$$

A computing system is the collection of all processing system that are under the same management. It has a name, a description, can be located, and provides some description for each of the collected processing systems.

$$\Psi^c = \{\psi^c_{1..n}\}, \psi^c_j = \langle n, d, \check{l}, \{\langle \psi^p, d \rangle\} \rangle$$

A facility (also known as a layer in computer networks) groups applications of the same scope. For instance, the network layer will group all applications with the scope of a network. In common taxonomies, BSS and OSS might be seen as layers, grouping business and operation support applications.

$$L = \{l_{1..n}\}, l_j = \langle n, d, \{\langle N^{A^p}, \psi^p \rangle\} \rangle$$

A domain group application that share a capability. It is often used to group applications under the same management as well.

$$\Delta = \{\delta_{1..n}\}, \delta_j = \langle n, d, N^{A^p} \rangle$$

With this application model we can now blueprint how an engine, in particular an intent engine, can be designed. Figure 6.12 shows the UML class diagram for an execution unit. Using the composite pattern, we can create a hierarchy of clusters, each may contain engines, and each engine may contain executors.

The smallest unit is an executor. It executes a piece of logic (for processing an intent). An executor x is a unit, so it comes with all elements of a unit as described above. Commonly, one would implement a generic intent executor realising the continuous process, and then run one of them per intent.

$$x \in U = \langle n, d, U^{\hat{l}}, I^x F^u \cup \Xi^x \rangle$$

An engine is an application that comes with the elements of applications as described above, plus a set of executors X and a set of language execution units $U^{\hat{l}}$ (as strategies to run executors).

$$x^e \in A = \langle n, d, \pi, U^i, U^a \cup X \cup U^{\hat{l}} \rangle$$

A cluster groups engines and clusters (recursively). It is a distributed application, with a set of supported engines X^e and a set of clusters X^c (full hierarchy).

$$x^c \in A^d = \langle n, d, \pi, U^i, U^d \cup X^e \cup X^c \rangle$$

An intent provider then is either an application or a distributed application. Since the intent process is broken down into tasks, and those tasks can be implemented by application units, we can distribute any functionality of an intent provider where it suits best: fully inside an engine, a proxy in the engine with standard or proprietary access to resources and/or other applications, or fully separated from the engine. Each provider can use its own implementation scenario.

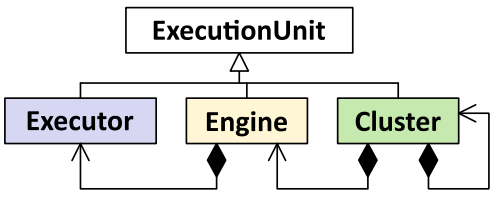


Figure 6.12: Application model, execution unit (UML).

6.2.4 Intent language

The intent language captures semantics of an intent declaration and expresses that in a syntax. An abstract syntax can be used to facilitate many concrete syntaxes, as required by interfaces (UI, CLI, etc.), different domains, and different providers. The main elements of an intent declaration are

- **Header** – a header with descriptive and administrative information about the intent. This includes an intent name (also known as identifier), description, the name of its creator, the template used to create it (if any), and a timestamp marking the creation.
- **Primary** – the primary intent objective as "who" wants "what". "who" is a name and a namespace in which the name can be resolved. For example, in an OSS there will be a list of users with identifiers and profiles. Here, the "who" can be declared as ns/user, with "ns" being the namespace (OSS users) and the "user" being a user ID. The "what" contains a verb (the required action) and an object (for the action). For instance, to count the lines of code of a product, this would state "count/lines+of+code" (note: the plus signs are used to allow for easier tokenising in a parser, for user interfaces they can be translated to actual white spaces).
- **Secondary** – the set (including the empty set) of secondary objectives. An intent can use the 5W1H categories to express any possible objective: when (time and time aspects), where (location or distance in some coordinate system), how (in what shape of form), why (optionally a reason for the objective, this might be helpful in negotiation processes and to understand priorities and urgencies), who (who else might be involved or excluded), what (what else might be involved or excluded).

An intent can then be written as an object containing three hash maps, one per element. The order of the map entries should not be important. For the header and the primary objective, these hash maps are flat (i.e. simple key/value lists or arrays). The map for secondary objectives may contain other hash maps. In simple concrete syntaxes this can be realised by overloading the key (e.g. using a '/' slash to create key hierarchies, effectively implementing a flat representation of a tree). In sophisticated concrete syntaxes associative arrays or map/tree object can be used.

An intent declaration can be translated from the abstract syntax to any concrete syntax. If a concrete syntax keeps the original semantics, then translation between those concrete syntaxes is also possible. If a concrete syntax does not keep the original semantics, e.g. when translated to some natural language, then a translation back to any other concrete or the original abstract syntax will not be possible.

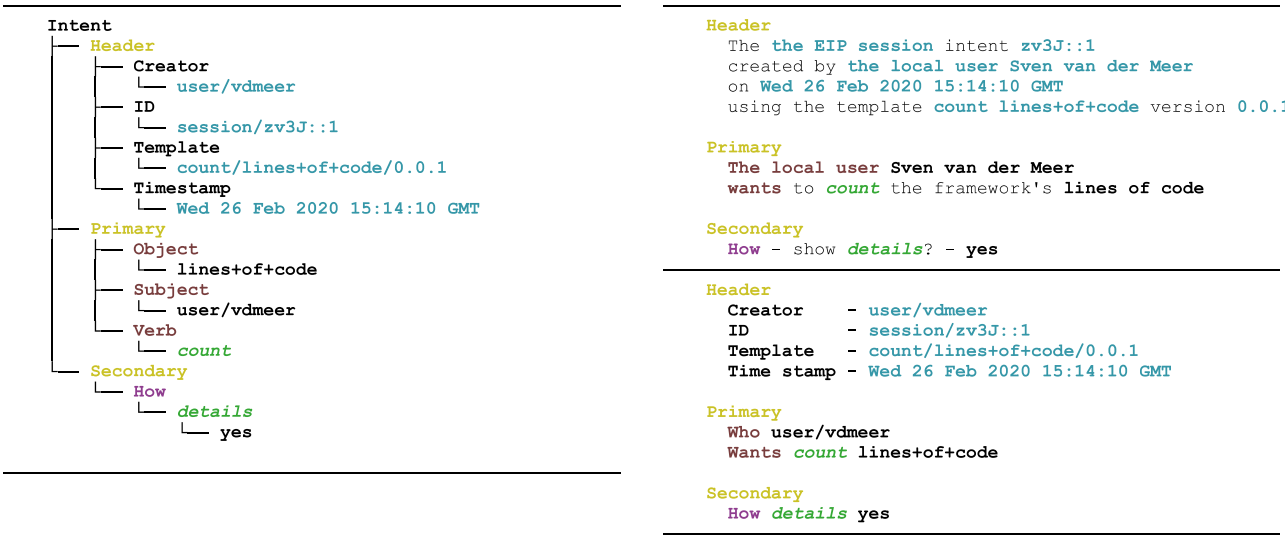


Figure 6.13: Intent language, abstract syntax (left) and translations (NLP and structured, right).

Figure 6.13 shows the abstract syntax (left side) and two translations into natural language and pseudo natural language (right side) for the example of "counting lines of code" used above. The intent was created by a user "vdmeer" in the namespace "users", which can be mapped to UNIX uses (or OSS users). The template identifier is associated with a session, which can be logged and archived for traceability. Template

and timestamp are also noted in the header.

The primary element then contains the subject (who, here a user named “vdmeer”), the verb (what is wanted, here “count”) and on which object (here “lines of code”). The secondary element contains one objective, namely that some “details” are required. This intent declaration can be easily (i.e. with static translation rules) translated into natural language and some sudo language. Since parts of the semantics are lost (here obfuscated by extra words, characters, and whitespaces), a translation back into the abstract syntax is impossible or very hard to achieve.

The same intent can be represented in many concrete syntaxes, some are shown in Figure 6.14, Figure 6.15, and Figure 6.16.

Figure 6.14 shows tree representations as JSON (left) and XML (right). These representations are the same as produced by the UNIX command “tree”. The underlying schema uses directories and files, where a directory (and the path to it) represents a key and the file name represents the content.

Figure 6.15 shows two different representations using JSON (left) and YAML (right). The first row is a direct translation from the abstract syntax into associative arrays. Some content is translated as string, which can be translated back to the original syntax. The second row shows JSON and YAML with the intent plus required information to create APEX events. APEX is a policy engine in the ONAP policy framework. It accepts these events and then triggers a policy (if deployed) to deal with the content. A simple APEX policy implementation can then be used to realise intents using policies.

Figure 6.16 shows a translation to single lines with tokens, forming a context-free language (left side). The right side of the figure shows the internal representation of the intent as a set of BASH 4 associative arrays (one per intent element header, primary, and secondary).

```
[
  {"type":"directory","name":"Intent","contents":[
    {"type":"directory","name":"Header","contents":[
      {"type":"directory","name":"Creator","contents":[
        {"type":"file","name":"user/vdmeer"}
      ]},
      {"type":"directory","name":"ID","contents":[
        {"type":"file","name":"session/zv3J::1"}
      ]},
      {"type":"directory","name":"Template","contents":[
        {"type":"file","name":"count/lines+of+code/0.0.1"}
      ]},
      {"type":"directory","name":"Timestamp","contents":[
        {"type":"file","name":"Wed 26 Feb 2020 15:14:10 GMT"}
      ]}
    ]},
    {"type":"directory","name":"Primary","contents":[
      {"type":"directory","name":"Object","contents":[
        {"type":"file","name":"lines+of+code"}
      ]},
      {"type":"directory","name":"Subject","contents":[
        {"type":"file","name":"user/vdmeer"}
      ]},
      {"type":"directory","name":"Verb","contents":[
        {"type":"file","name":"count"}
      ]}
    ]},
    {"type":"directory","name":"Secondary","contents":[
      {"type":"directory","name":"How","contents":[
        {"type":"directory","name":"details","contents":[
          {"type":"file","name":"yes"}
        ]}
      ]}
    ]}
  ]}
]
```

```
<?xml version="1.0" encoding="UTF-8"?>
<tree>
  <directory name="Intent">
    <directory name="Header">
      <directory name="Creator">
        <file name="user/vdmeer"></file>
      </directory>
      <directory name="ID">
        <file name="session/zv3J::1"></file>
      </directory>
      <directory name="Template">
        <file name="count/lines+of+code/0.0.1"></file>
      </directory>
      <directory name="Timestamp">
        <file name="Wed 26 Feb 2020 15:14:10 GMT"></file>
      </directory>
    </directory>
    <directory name="Primary">
      <directory name="Object">
        <file name="lines+of+code"></file>
      </directory>
      <directory name="Subject">
        <file name="user/vdmeer"></file>
      </directory>
      <directory name="Verb">
        <file name="count"></file>
      </directory>
    </directory>
    <directory name="Secondary">
      <directory name="How">
        <directory name="details">
          <file name="yes"></file>
        </directory>
      </directory>
    </directory>
  </directory>
</tree>
```

Figure 6.14: Intent Language, formal trees in JSON (left) and XML (right).

```
{
    "Intent" : {
        "Header" : {
            "Creator" : "user/vdmeer",
            "ID" : "session/zv3J::1",
            "Timestamp" : "Wed 26 Feb 2020 15:14:10 GMT",
            "Template" : "count/lines+of+code/0.0.1"
        },
        "Primary" : {
            "Subject" : "user/vdmeer",
            "Verb" : "count",
            "Object" : "lines+of+code"
        },
        "Secondary" : [
            "How details yes"
        ]
    }
}
```

```
---
Intent
    Header
        Creator:user/vdmeer
        ID:session/zv3J::1
        Timestamp:Wed 26 Feb 2020 15:14:10 GMT
        Template:count/lines+of+code/0.0.1
    Primary
        Subject:user/vdmeer
        Verb:count
        Object:lines+of+code
    Secondary:
        - How details yes
...
```

```
{
    "nameSpace" : "%%NAME_SPACE%%",
    "name" : "%%EVENT_NAME%%",
    "version" : "%%EVENT_VERSION%%",
    "source" : "%%EVENT_SOURCE%%",
    "target" : "%%EVENT_TARGET%%",
    "Intent" : {
        "Header" : {
            "Creator" : "user/vdmeer",
            "ID" : "session/zv3J::1",
            "Timestamp" : "Wed 26 Feb 2020 15:14:10 GMT",
            "Template" : "count/lines+of+code/0.0.1"
        },
        "Primary" : {
            "Subject" : "user/vdmeer",
            "Verb" : "count",
            "Object" : "lines+of+code"
        },
        "Secondary" : [
            "How details yes"
        ]
    }
}
```

```
---
nameSpace:%%NAME_SPACE%%
name:%%EVENT_NAME%%
version:%%EVENT_VERSION%%
source:%%EVENT_SOURCE%%
target:%%EVENT_TARGET%%
Intent
    Header
        Creator:user/vdmeer
        ID:session/zv3J::1
        Timestamp:Wed 26 Feb 2020 15:14:10 GMT
        Template:count/lines+of+code/0.0.1
    Primary
        Subject:user/vdmeer
        Verb:count
        Object:lines+of+code
    Secondary:
        - How details yes
...
```

Figure 6.15: Intent language, JSON (left) and YAML (right) versions.

```
Intent Header Creator user vdmeer
Intent Header ID session zv3J::1
Intent Header Template count lines+of+code 0.0.1
Intent Header Timestamp "Wed 26 Feb 2020 15:14:10 GMT"

Intent Primary Who user/vdmeer
Intent Primary Wants count lines+of+code

Intent Secondary How details yes
```

```
INTENT_HEADER[creator]    = user/vdmeer
INTENT_HEADER[id]         = session/zv3J::1
INTENT_HEADER[template]   = count/lines+of+code/0.0.1
INTENT_HEADER[timestamp] = Wed 26 Feb 2020 15:14:10 GMT

INTENT_PRIMARY[subject] = user/vdmeer
INTENT_PRIMARY[verb]    = count
INTENT_PRIMARY[object]  = lines+of+code

INTENT_SECONDARY[how/details] = yes
```

Figure 6.16: Intent language, translation (line, left) and flat arrays (BASH, right).

All figures shown in this subsection show the same intent (count lines of code with details). Any other intent can be translated into the same concrete syntaxes and formats. The only difference then are the details of primary and secondary objectives, and the potential length of secondary objectives (since they can declare a long list of items).

6.3 Modelling 5G-CLARITY use cases as intents

In this section we explore how to model using intents several management operations in 5G-CLARITY, including the operations required by some of the ML models introduced in Section 4 which are instantiated in the AI engine, and some of the slice provisioning operations enabled by the 5G-CLARITY management and orchestration stratum. Additional use cases may be modelled using intents in subsequent deliverables.

Each use case is described using a summary table. At the current modelling stage, the focus is on the main objective, the required input (later to be modelled as intent) and the provided output of the algorithm along with an action (later to be modelled as an intent).

- **Title**: describes the primary objective of the algorithm, which indicates or leads directly to the primary part of an intent the algorithm does provide
- **Input**: Describing Parameters and semantics of the input required by the algorithm. This input will be modelled later as an intent (or parts of an intent) in form of a variation of secondary objectives (i.e. parameter with given semantics and a required or expected value).
- **Output**: Describes the output the algorithm produces, with parameters and defined semantics. This output can later be used as part of an intent, here variations of secondary objectives with semantics and values.
- **Intent Giver**: Determines the role that should create (or give) an intent to the algorithm. In most cases, this will be an operator using the dashboard, in some cases (especially for future algorithms), this might also be another component of a 5G system.
- **Intent action**: Defines the decided action in terms of what the algorithm states a system to realise. This action will later be aligned with other intent providers. Here it represents most likely the primary objective. For instance, an algorithm might decide to activate parts of a Wi-Fi network, then the action would be "activate Wi-Fi" (primary) with more details (which parts, secondary). This intent can then be used on any provider able to activate a Wi-Fi network.
- **Intent Provider**: Describing the entity in the 5G-CLARITY system in charge of executing the intent action. Using this statement, we can design the required providers which abstract from other 5G Clarity components in other planes of the 5G network.

Complementing the previous table each use case includes a sequence chart describing the interactions between the intent giver, the intent engine and the intent providers.

6.3.1 SLA violation-success rate prediction

This use case corresponds to the ML algorithm described in Section 4.2.

Within the 5G-CLARITY management architecture, the SLA violation/success rate prediction process follows the flow of operations depicted in Figure 6.17 and summarized as follows:

- Step 1: the private network operator sends a request to the intent engine on the probability/margin of possible SLA violations/success rate which is based on a basic natural (English) language such as "I want a prediction on SLA violations". This intent is summarized in Table 6-1.
- Step 2: the intent engine translates this request to ML model level and forwards this request to the ML model.
- Step 3: the ML model decides what type of telemetry data is needed to provide a prediction on the SLA violations such as (i) UE aggregated throughput that is reported by gNB/AP to understand traffic volume; (ii) offered load per cell to understand spatial volume distribution; (iii) Slice/Service Type (SST) and success rate to understand traffic class and its performance; (iv) UE connected cell information to understand user mobility pattern; and (v) the SLA details to obtain the violation/success rate. Then, the ML model sends a request to intent engine on providing the required telemetry data, as outlined in Table 6-1.
- Step 4: the intent engine translates this request to the telemetry collector.
- Step 5: the telemetry collector provides the requested data to the intent engine.
- Step 6: the intent engine provides the data to the ML model.

- Step 7: the telemetry data is firstly used by the offline trained ML model to forecast the network traffic/load for the overall network traffic volume and spatial volume distribution. Then the estimated network traffic/load is used to predict possible SLA violations/success rate which is then sent to the intent engine as a probability/margin for possible SLA violations/success rate.
- Step 8: the intent engine translates the estimated probability/margin for possible SLA violations to basic natural language and informs the private network operator about the margins.

Table 6-1. SLA Violation/Success Rate Prediction – Operator to the ML Model.

Intent 1: SLA violation/success rate prediction

Input	Details of the SLA terms
Output	Configure and trigger ML model for network traffic/load forecast in order to predict the overall network traffic volume and spatial volume distribution, SLA violation/success rate prediction
Intent Giver	Operator (human intervention), AI engine (closed loop)
Intent Action	Compute a probability/margin for possible SLA violations/success rate
Intent Provider	ML model of network traffic/load

Table 6-2. SLA Violation/Success rate Prediction – ML Model to Telemetry.

Intent 2: Telemetry data request and ML model computation

Input	Telemetry data request (UE aggregated throughput, load per cell, Slice/Service Type (SST) and success rate, UE connected cell
Output	SLA violation/success rate prediction
Intent Giver	ML model
Intent Action	Provide telemetry data from network to the ML model and compute SLA violation/success rate prediction
Intent Provider	Data Processing subsystem

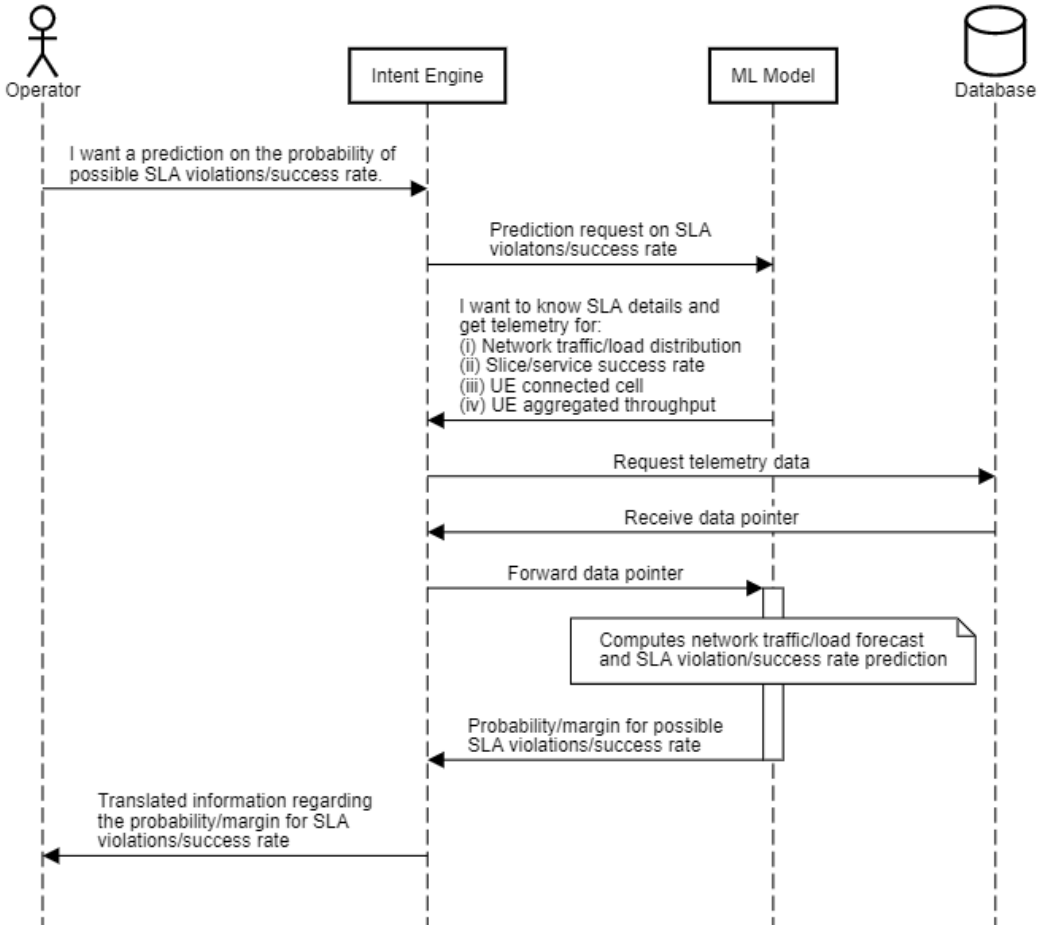


Figure 6.17. Flow of SLA violation/success rate prediction process within 5G-CLARITY management architecture.

6.3.2 AT3S traffic routing/handover

This use case corresponds to the ML algorithm described in Section 4.3.

The following intents are identified:

- Intent 1 (Table 6-3): Used by the NPN operator to set up a high-level business policy demanding high reliability for a specific connection, e.g. a particular device in the network.
- Intent 2 (Table 6-4): Intent generated by the ML model to request a subset of Telemetry data to the 5G-CLARITY data processing subsystem (see Section 2.4.2).
- Intent 3 (Table 6-5): Based on the received telemetry the ML model computes the desired AT3S policy configuration and an intent is generated to enforce such configuration.

Table 6-3: Maintenance of Link reliability – Operator to the ML Model.

Intent 1: Maintenance of link reliability

Input	Set of goals provided by the operator on reliability performance/requirement
Output	Configure and trigger the ML model that models UE mobility and routes traffic flows in order to achieve link reliability requirements
Intent Giver	Operator (human intervention), AI engine (closed loop)
Intent Action	Configure and trigger the ML model to decrease packet drop rate

Intent Provider	ML model for UE mobility prediction and traffic routing.

Table 6-4: Maintenance of Link Reliability -- ML Model to Telemetry.

Intent 2: Telemetry data request and computation of ML model

Input	Telemetry data for UE connected cell/AP/SSID (user mobility pattern), UE DL packet drop rate (handover failure), RSSI (link performance/blockage)
Output	Pointer to required telemetry
Intent Giver	ML model/algorithm
Intent Action	Request data to the Data Processing subsytem
Intent Provider	Data processing subsystem

Table 6-5: Maintenance of Link Reliability -- ML Model to Telemetry.

Intent 2: Telemetry data request and computation of ML model

Input	AT3S weights for a given UE
Output	Configuration of AT3S user plane function
Intent Giver	ML model/algorithm
Intent Action	Configure the AT3S user plane scheduling weights in the user plane function managing the considered UE
Intent Provider	AT3S controller in RIC

A sequence chart describing the previous intents and involved elements is depicted in Figure 6.18.

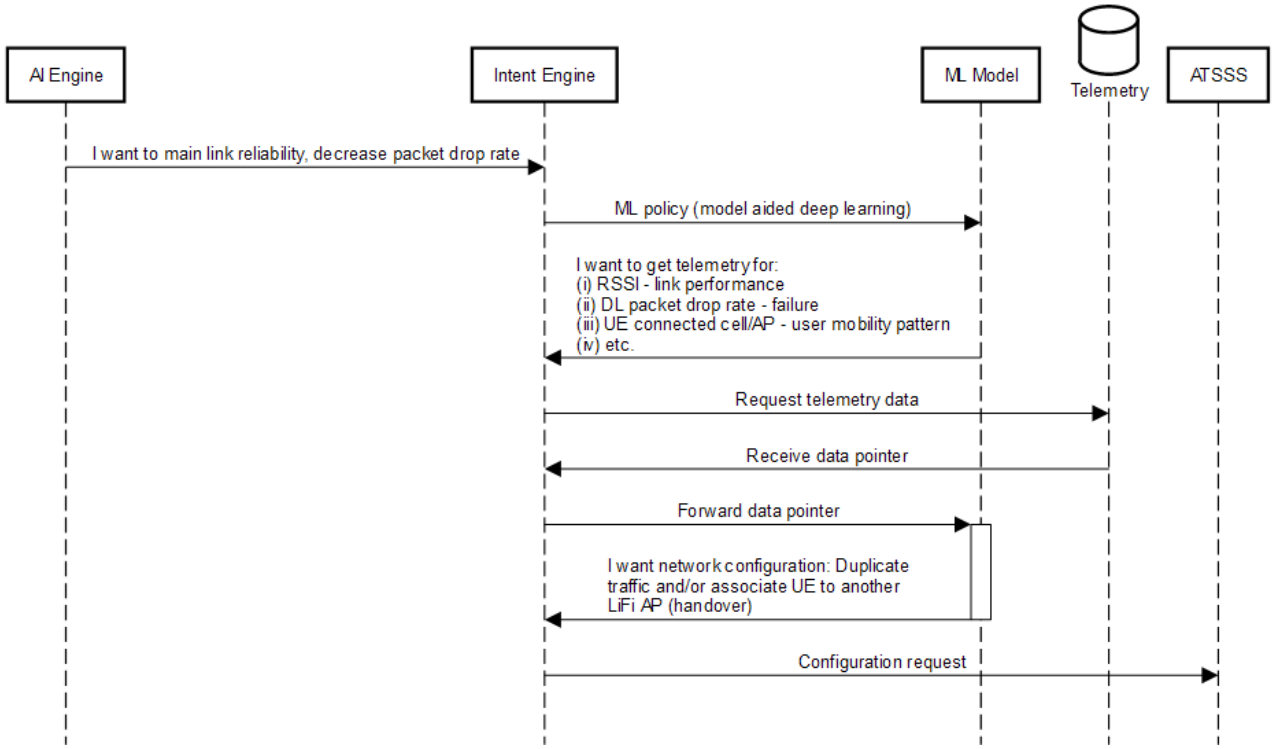


Figure 6.18. Sequence diagram for the AT3S ML model interaction with the Intent Engine

6.3.3 Resource provisioning and optimisation

This use case corresponds to the ML algorithm described in Section 4.8, and the required intents are described in Table 6-6.

Table 6-6 Specification of Intents Required for Resource Provisioning and Optimization.

Intent 1: Required slice SLA	
Input	Performance attributes (e.g. required capacity), business intents (i.e. operator's high-level goals)
Output	Configure and instantiate ML model to enforce desired policy
Intent Giver	NPN Operator (human intervention), AI engine (closed loop)
Intent Action	Instantiate required ML model in AI Engine
Intent Provider	AI Engine
Intent 2: Telemetry provisioning	
Input	Metrics related to network performance monitoring (e.g. aggregated throughput, resource utilisation and number of active users) that are required by the ML model
Output	Requested metrics with their associated values
Intent Giver	AI engine (ML model)
Intent Action	Provide telemetry to the AI engine
Intent Provider	Telemetry Collector
Intent 3: Update controller's configuration	
Input	Rule(s) of the non-real time controller modifying network parameters related to 5GNR/LTE (e.g. transmit power, bandwidth), Wi-Fi (e.g. transmit power, channels), LiFi (TBD) and others (e.g. AT3S related)
Output	Updated configuration of the controller
Intent Giver	AI engine (ML model)
Intent Action	Update on multi-WAT non-RT parameters
Intent Provider	Slice Manager

The resource provisioning and optimization process consists of a set of operations that are described in Figure 6.19. In particular, the private network operator may send a request (Intent 1 - Table 6-6) towards the Intent Engine in order to deploy a 5G-CLARITY slice. This request is based on natural (English) language such as "I want a 5G-CLARITY slice" and it can include performance and functional attributes related to the desired behavior of the slice. Alternatively, the private network operator may send a business intent that is related to its high-level objectives, e.g. "I want to minimize energy consumption".

The Intent Engine delivers this request together with additional information (e.g. SLAs, list of business intents) to the AI Engine. This module decides what telemetry is required to perform the resource provisioning and optimization. As a result, it generates a new intent (Intent 2 - Table 6-6) towards the Intent Engine in order to request telemetry (e.g. aggregated throughput, resource utilization levels, number of active users, etc.). After processing the intent, the telemetry module receives the data request and it provides the information required to access the data (e.g. a data pointer). This information is delivered to the AI Engine through the Intent Engine. Once the ML model receives the telemetry, it is able to derive the optimal configuration of the non-real time controller that is in charge of resource provisioning. Then, such configuration will be sent

in the form of an intent to the Slice Manager via the Intent Engine (Intent 3 - Table 6-6).

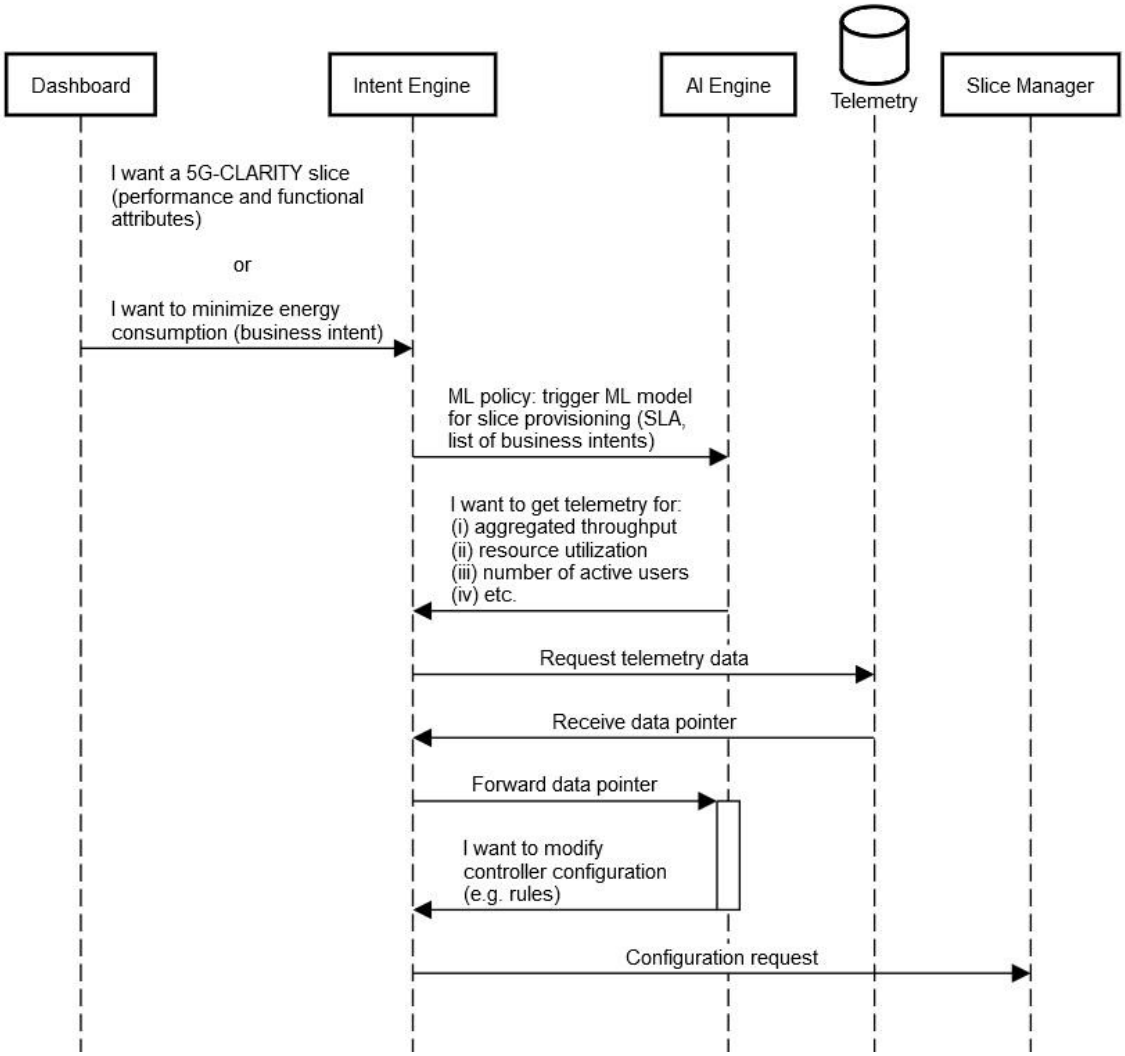


Figure 6.19 Flow of resource provisioning and optimization process within 5G-CLARITY management architecture.

6.3.4 RAN slicing in multi-tenant networks

This use case corresponds to the ML algorithm described in Section 4.4. Figure 6.20 shows the workflow associated to this use case including the different interactions with the intent engine.

During the whole process three intents are generated towards the intent engine. The first intent (see Table 6-7) is triggered by the operator, requesting a RAN slice for a tenant with specific SLA terms given by an aggregate capacity across all cells and a maximum bit rate per cell. In order to fulfil this request, the intent engine will inform the ML model about the SLA terms of this RAN slice so that it can determine the resource quota assigned to the different tenants in each cell. For this purpose, the ML model will trigger a new intent (intent 2, detailed in Table 6-8), requesting the required measurements by the algorithm, which are the resource usage (i.e. the fraction of physical resources used for data traffic in a cell) per cell and tenant, the total throughput per tenant across all the cells and the offered load per cell and tenant. Then, the intent engine will contact the telemetry system, which will gather the measurements and will provide them back to the intent engine that will forward them to the ML model. Based on the received information, the ML model will determine the resource quota to be assigned to each tenant in each cell and will provide the result to the slice manager through another intent (intent 3, detailed in Table 6-9), so that it can enforce this configuration in the network.

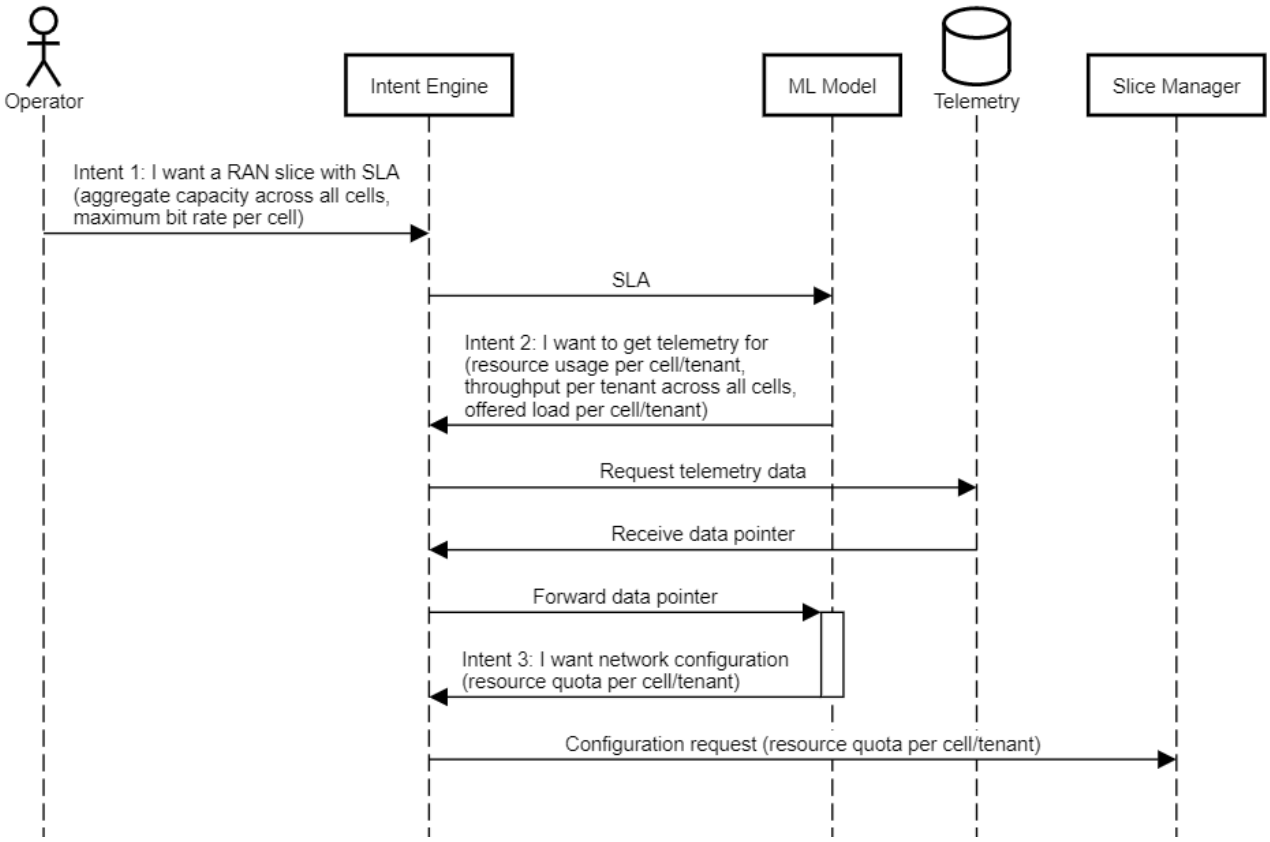


Figure 6.20 Workflow of the RAN slicing for multi-tenant networks use case.

Table 6-7 Intent 1 in the ML Algorithm for RAN Slicing in Multi-Tenant Networks.

Intent 1: Provide SLA for each slice

Input	SLA terms (aggregate capacity across all cells, maximum bit rate per cell) for the RAN slice of a tenant
Output	Configure and trigger ML model in AI engine
Intent Giver	Operator
Intent Action	To contact the ML model to determine the configuration of the resource quota assigned to each tenant in each cell
Intent Provider	ML model

Table 6-8 Intent 2 in the ML Algorithm for RAN Slicing in Multi-Tenant Networks.

Intent 2: Obtain Telemetry

Input	Telemetry data request (resource usage per cell and tenant, total throughput per tenant across all the cells and offered load per cell and tenant)
Output	Pointer to the requested telemetry data
Intent Giver	ML model
Intent Action	Gather telemetry data from the network
Intent Provider	Data processing subsystem

Table 6-9 Intent 3 in the ML Algorithm for RAN Slicing in Multi-Tenant Networks.

Intent 3: Update resource allocation in each cell/tenant	
Input	Resource quota per cell/tenant to be configured
Output	Configuration of the resource quota
Intent Giver	ML model
Intent Action	Configure the assigned resource quota per cell/tenant
Intent Provider	Slice Manager

6.3.5 Dynamic transport network setup and computing resource provisioning

This use case corresponds to the ML algorithm described in Section 4.9, and comprises the following intents:

- Intent 1 (Table 6-10): The operator communicates a high-level policy to the Intent Engine. Then, the involved ML models are configured to be driven by the operator intents and launched. The operator intents might be both high level business goals (e.g., maximize the network revenue) or performance objective (e.g., minimize the energy consumption of the network).
- Intent 2 (Table 6-11): The ML algorithm request the required telemetry and predictive data analytics to compute the configuration of the TN and the computing resources allocation for the network services. The requested data provides a summary of the network state, which represents the observations of the ML models.
- Intent 3 (Table 6-12): The ML algorithm issues a command to configure the transport network.
- Intent 4 (Table 6-13): The ML algorithm issues a command to allocate or release computing resources to the network services

Table 6-10: Intent to Configure and Trigger the ML Models for Dynamic Transport Network Setup and Computing Resources Provisioning.

Intent 1: Transport network policy provisioning	
Input	Set of goals provided by the operator to drive the network operation.
Output	Configuration and triggering of the required ML models to achieve the operator's objectives.
Intent Giver	Operator.
Intent Action	Configure and trigger the needed ML models to optimize the dynamic transport network setup and computing resources provisioning in order to meet the operator's goals.
Intent Provider	ML models.

Table 6-11: Intent to Request Telemetry Data and Predictive Data Analytics.

Intent 2: Telemetry request	
Input	Set of telemetry metrics (e.g., servers and links resources utilization) and data analytics (e.g., expected traffic matrix, nodes and links time-to-failures).
Output	Value of the requested telemetry metrics and data analytics.
Intent Giver	ML models (AI Engine)
Intent Action	Provide ML models with access to the requested telemetry metrics and data analytics.
Intent Provider	Telemetry and AI Engine. The latter hosts ML models to realize the predictive data analytics.

Table 6-12: Intent to Configure the Transport Network.

Intent 3: Transport network configuration	
Input	- Links time-to-failures - Temporal profile demand per 5G-CLARITY slice, per 5QI, and per source/destination pair (Predicted traffic matrix). - Traffic characteristics per 5QI (e.g., moments for the data rate process) - Network topology - Links resources utilization.
Output	- The translation of the 5QIs into IEEE 802.1Q traffic classes for each 5G-CLARITY slice - Aggregated transport resources (e.g., link capacities and buffer sizes) per traffic class and per 5G-CLARITY slice - Paths allocated for each 5G-CLARITY slice - Output port configuration of the TSN bridges, e.g., gate control list and time windows size
Intent Giver	ML models (AI Engine)
Intent Action	Configure the transport network as specified in the request.
Intent Provider	SDN Controller.

Table 6-13: Intent to Scale Network Services.

Intent 4: Compute resource allocation	
Input	- Computing nodes time-to-failures - Temporal profile demand per 5G-CLARITY slice, per 5QI, and per source/destination pair (Predicted traffic matrix). - Traffic characteristics per 5QI (e.g., moments for the data rate process) - Processing time distributions for every VNF/Service app - Clouds available resources.
Output	- Computing resources (e.g., memory, disk and CPU) to be allocated to each VNF instance. - Set of affinity rules that restrict the instantiation of the VNF instances on the servers.
Intent Giver	ML models (AI Engine).
Intent Action	Scale the specified set of network services: i) reserve/release computational resources, ii) apply the specified set of affinity rules to deploy the VNFs on the servers.
Intent Provider	NFVO.

The previous intents and the various actors involved are depicted in the sequence diagram of Figure 6.21: Operation call flow of the ML algorithm for the dynamic resource provisioning and the transport network configuration.

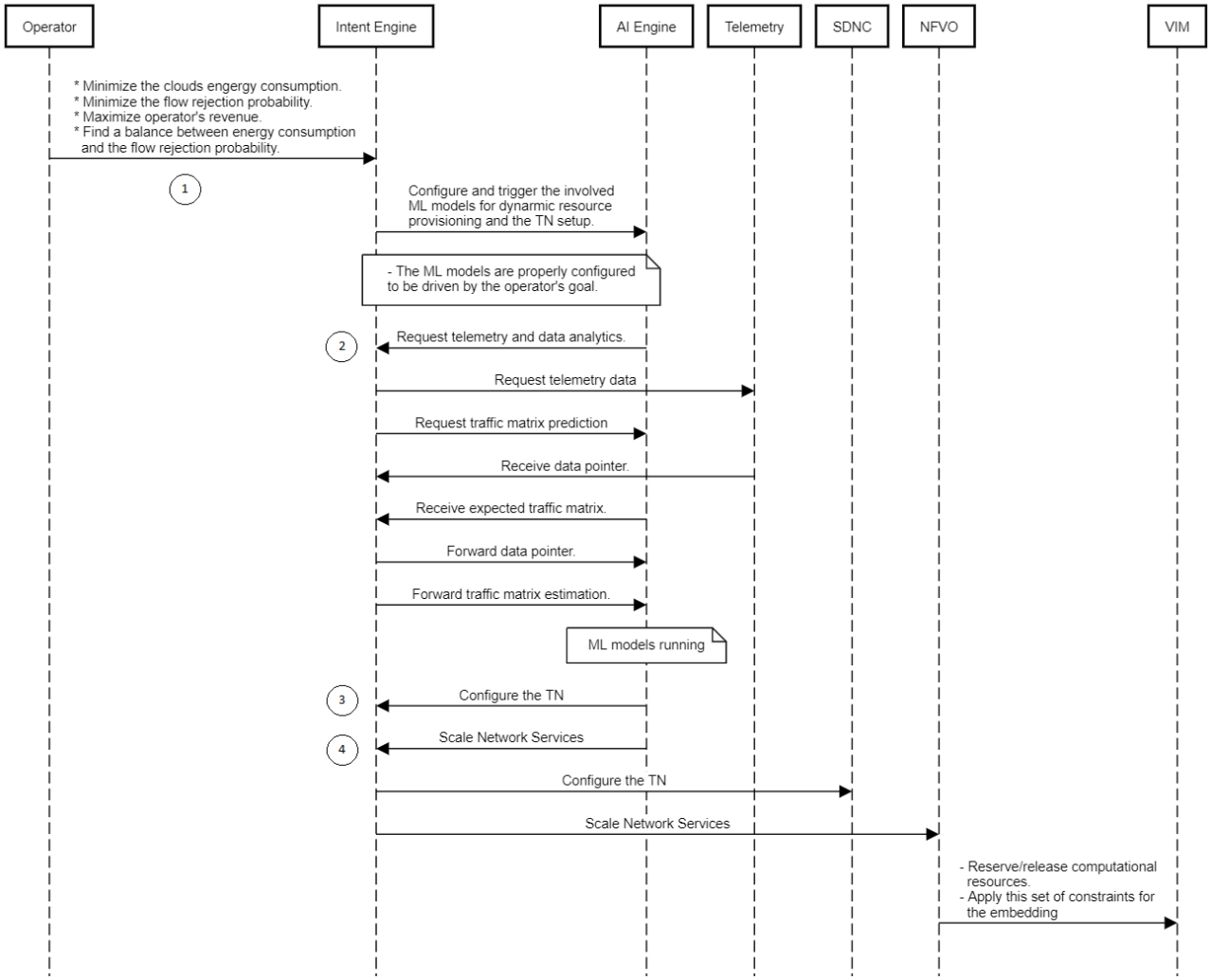


Figure 6.21: Operation call flow of the ML algorithm for the dynamic resource provisioning and the transport network configuration.

6.3.6 Intent-based slice provisioning

This use case describes an intent based slice provisioning workflow using the 5G-CLARITY slice and service provisioning system introduced in Section 2. Two basic intents are identified:

- Intent 1: Deploy a slice, which configures the 5G-CLARITY multi-tenant infrastructure to provision one slice.
- Intent 2: Instantiate a service, which deploys an application service in a given slice.

These intents are further described in Table 6-14.

Table 6-14: Slice provisioning process

Intent 1: 5G-CLARITY Slice provisioning	
Input	Infrastructure nodes participating in the slice: wireless and compute Quotas for the wireless, transport and compute nodes Slice Identifiers: PLMNIDs, S-NSSAIs, SSIDs
Output	A pointer to the deployed slice is returned

Intent Giver	NPN Operator
Intent Action	Provision a 5G-CLARITY slice
Intent Provider	Slice Manager

Intent 2: 5G-CLARITY Service Instantiation

Input	Slice identifier, service identifier
Output	Pointer to the running service
Intent Giver	NPN Operator or 3rd party (e.g. MNO)
Intent Action	Instantiate an application service
Intent Provider	Slice Manager (through NFVO)

The sequence chart in Figure 6.22 describes the interactions between the Intent giver (user), the Intent Engine and the Slice Manager that is the main provider for the slice and service provisioning intents. Notice however that many other components of the 5G-CLARITY management and orchestration stratum are hidden behind the slice manager (e.g. multi-WAT non-RT RIC, Transport SDN controller, NFVO) and not shown to simplify the diagram.

It is worth noting that the two intents described in this section admit variations, i.e.:

- Intent 1 to provision a slice, depends on whether a new slice is instantiated requiring a new PLMNID, or on whether the PLMNID is already in place and a S-NSSAI is deployed,
- Intent 2 to instantiate a service, depends on whether there is an already running service on that slice that needs to be replaced.

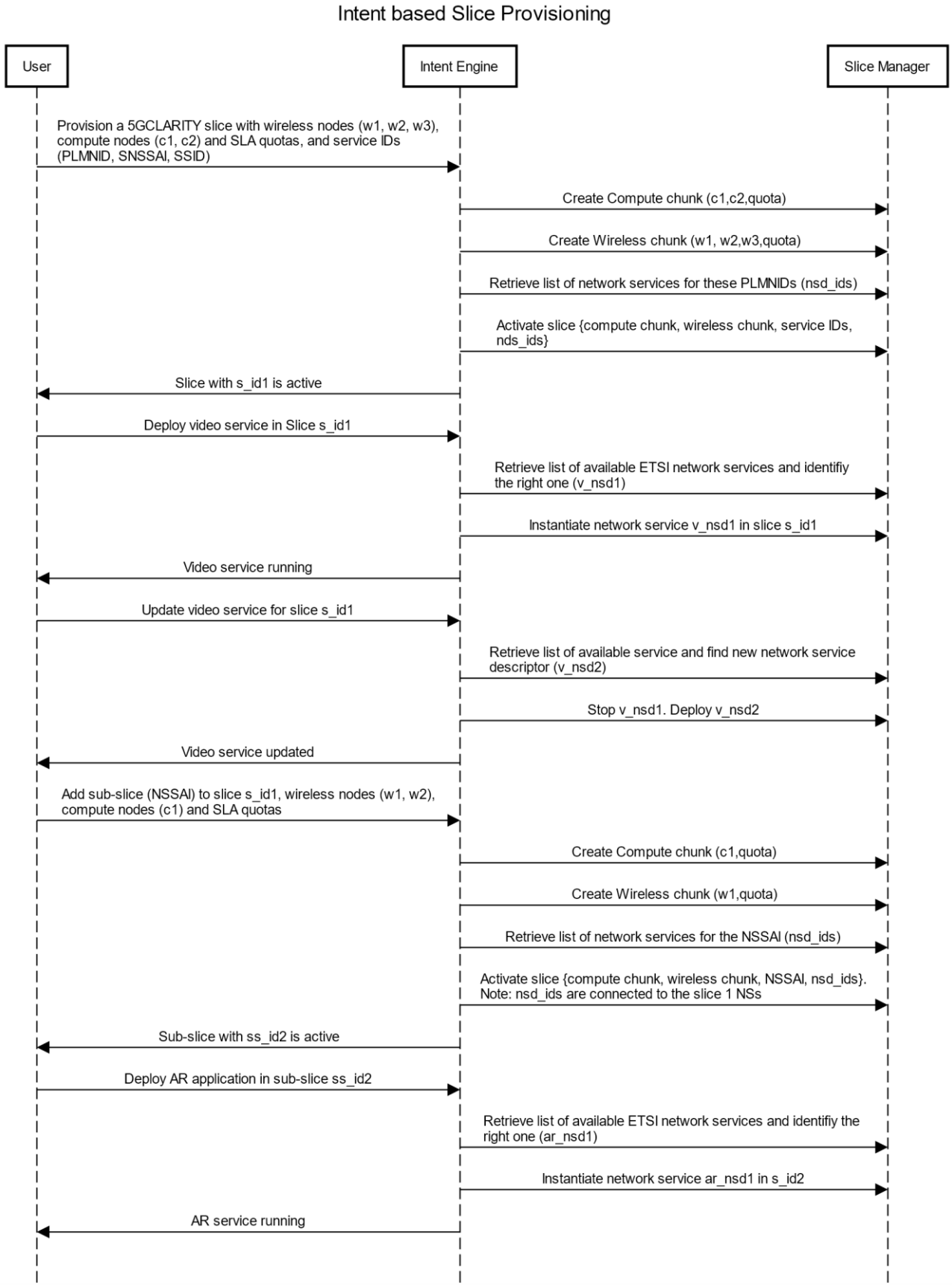


Figure 6.22. Intent based slice provisioning.

6.3.7 Adaptive defect detection a smart factory

This use case corresponds to the ML algorithm described in Section 4.10.

The operator or NPN smart factory worker provides adaptive defect detection intent to the Intent Engine (Intent 1 - Table 6-15) along with a set of goals that is going to be used to recognize potential defects in the production line. Then, the Intent Engine configures and initiates ML algorithm for adaptive defect detection that is responsible to detect defective products/pieces and to remove them from the production line.

Table 6-15 Intent to Configure and Trigger Defect Detection ML Model.

Intent 1: Defect detection monitoring	
Input	Set of goals provided by the operator/factory worker to recognize potential defects in the production line.
Output	Configure and trigger ML model for adaptive defect detection
Intent Giver	Operator/NPN smart factory worker
Intent Action	Trigger, configure, deploy and monitor the involved ML models according to the high-level goals provided of the operator's intent
Intent Provider	ML model

Once the ML model for defect detection is triggered, various telemetry data from sensors, UE modems, network nodes, cameras, applications etc. are requested from data lake via the Intent Engine (Intent 2 - Table 6-16). Intent Engine gets the information regarding the required telemetry from the AI engine and passes it to the data lake or telemetry database. Then, data lake processes this request and provides the required data to the AI engine which then uses the data to train/retrain the ML model, manage life cycle of the ML model and predict any defective piece/product in the production line.

Table 6-16 Intent to Request Telemetry Data and ML Model Configuration

Intent 2: Telemetry data request and ML model configuration	
Input	- UE modems and network nodes telemetry - Computing hosts telemetry, e.g. edge servers, CPU/GPU/FPGA characteristics - Object detection application telemetry, e.g. type of objects, defect conditions - Sensors and cameras telemetry - Targeted operation conditions including timing budget and energy budget
Output	- ML model in use - Allocated resources for hosting the ML model - Measurement data for monitoring and life cycle management of the ML model - Inference/prediction actions towards the physical controller equipment in the factory, e.g. robotic arm, conveyor belt, camera
Intent Giver	ML model/AI engine
Intent Action	Provide the requested telemetry to the ML model
Intent Provider	Telemetry/data lake and ML model/AI engine

Based on the output of the ML model for defect detection, i.e. if a defective piece/product is detected, the Intent Engine (Intent 3 - Table 6-17) either (i) alerts the smart factory worker to command an intervention; or (ii) automatically command a configuration/action to the physical controller equipment such as robots, robotic arm, conveyor belt, etc. to stop the line or remove the defective product from the production line.

Table 6-17 Intent to Remove Defective Product/Piece from the Production Line

Intent 3: Intervention/removal of the defect	
Input	Inference/defect detection result
Output	If a defect is detected, either: - Alert a remote worker to command an intervention (human intervention) - Command a configuration/action towards the physical controller equipment in the factory, e.g. robotic arm, conveyor belt, camera (adaptive automation)
Intent Giver	ML model/AI engine (closed loop)
Intent Action	If there is a defective product/piece, stop the line or take the defective piece out of the production line
Intent Provider	Smart factory worker (human intervention)/controller equipment (adaptive automation)

The sequence chart in Figure 6.23 depicts the previous intents and elements involved.

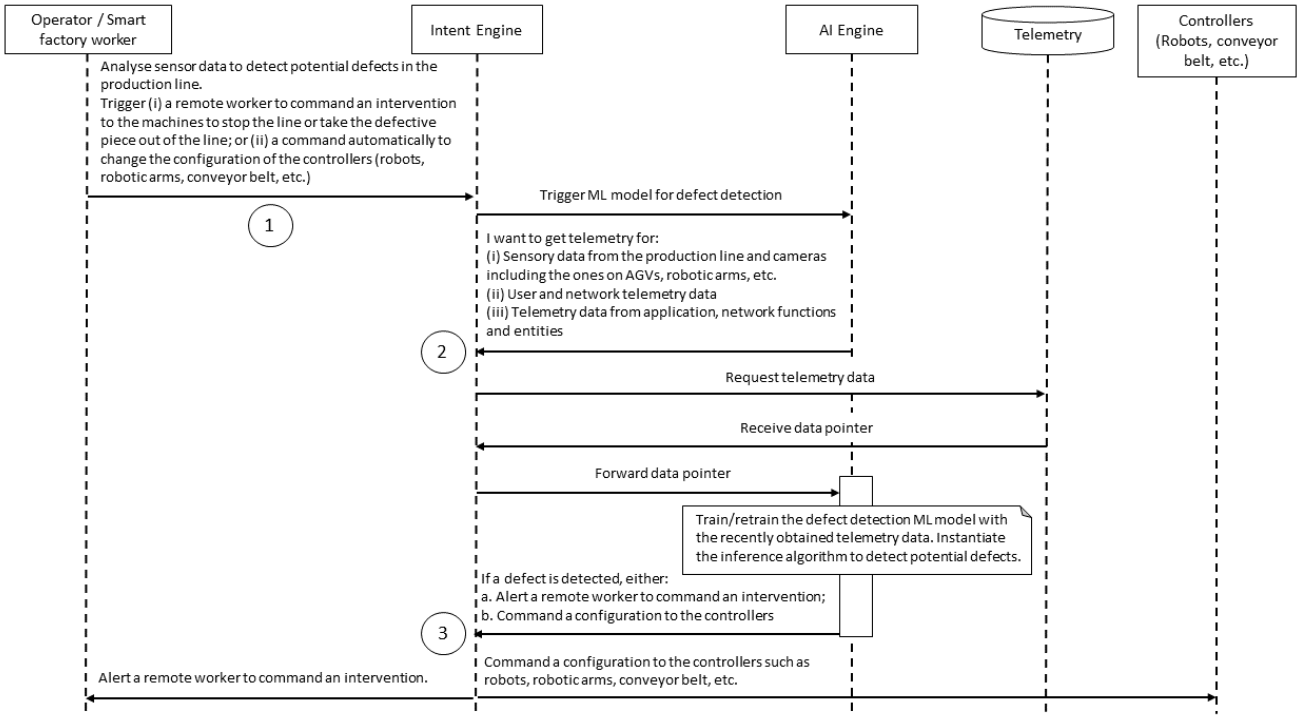


Figure 6.23 Flow of adaptive AI-based defect detection in a smart factory.

7 Conclusions and Next Steps

This deliverable has specified the details of the parts of 5G-CLARITY architecture that realize the initial design of the SDN/NFV platform and identification of target 5G-CLARITY ML algorithms including, *1)* network slicing solution for private venues; *2)* integrated multi-WAT real-time telemetry system; *3)* integration of private and public networks; *4)* ML algorithms for supporting autonomous network management; *5)* AI-engine that deals with the execution and maintenance of ML models; and *6)* intent-based networking for facilitating customer interaction in private networks. These are critical functionalities for satisfying the requirements 5G use cases, especially where the AI play a crucial role, and their implementation in the next phase of the 5G-CLARITY project should lead to important extensions of the state of the art solutions for management, orchestration and intelligence stratums.

The design of related components, which is detailed in Section 2 to 6 has addressed the main objectives that were identified in the introduction. Section 1.2 has demonstrated the initial design and requirements of 5G-CLARITY management and orchestration stratum and intelligence stratum. Section 2 provided the initial design for the multi-WAT non-RT Controller and the Slicing solution in the private and public networks (from slice preparation to the configuration and NS provisioning). Section 3 described the type of management models in the 5G-CLARITY and integration of private network with the management system of public networks. Sections 4 and 5 focused on leveraging machine learning for supporting autonomous network management from different perspectives. The ML will be deployed in the 5G-CLARITY's AI engine as ML services, consuming data from the 5G-CLARITY Data Management component and provide predictions for 5G-CLARITY management and network functions. Finally, Section 6 has given design of intent-based policy language, which will be sat over the slice manager and used by the AI engine to ease the management and operation of the network.

Next for 5G-CLARITY is to implement and develop, *1)* the initial slicing models in conjunction of MNOs with the private network slices; *2)* AI engine with 9 machine learning algorithms to perform a set of network management functions in 5G-CLARITY platform; and *3)* intent-based networking interface.

8 References

[1] 5G-CLARITY Deliverable D2.1, “Use Cases and Requirements”, March 2020.

[2] 5G-CLARITY Deliverable D2.2, “Primary System Architecture”, October 2020.

[3] 5G-CLARITY Deliverable D3.1, “State-of-the-Art Review and Initial Design of the Integrated 5GNR/Wi-Fi/LiFi Network Frameworks on Coexistence, Multi-Connectivity, Resource Management and Positioning”, August 2020.

[4] OpenStack https://www.openstack.org/

[5] libvirt: The virtualization API https://libvirt.org/

[6] Kubernetes https://kubernetes.io/

[7] Open Networking Foundation (ONF), OpenFlow Switch Specification Version 1.5.1, March 2015.

[8] M. Suñé, V. Alvarez, T. Jungel, U. Toseef and K. Pentikousis, "An OpenFlow Implementation for Network Processors," *2014 Third European Workshop on Software Defined Networks*, London, 2014, pp. 123-124, doi: 10.1109/EWSDN.2014.17.

[9] Haoyu Song. 2013. Protocol-oblivious forwarding: unleash the power of SDN through a future-proof forwarding plane. In *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking* (*HotSDN ’13*). Association for Computing Machinery, New York, NY, USA, 127–132. DOI:https://doi.org/10.1145/2491185.2491190

[10] Quentin Monnet, “OpenState: An interface for Stateful Packet Processing in Programmable Switches,” IEEE Softwarization, March 2017. [Online]. Available: https://sdn.ieee.org/newsletter/march-2017/openstate-an-interface-for-stateful-packet-processing-in-programmable-switches

[11] The P4 Language Consortium, “$P4_{16}$ Language Specification version 1.2.1”, June 2020. [Online]. Available: https://p4.org/p4-spec/docs/P4-16-v1.2.1.pdf

[12] B. Belter *et al.*, "Programmable Abstraction of Datapath," *2014 Third European Workshop on Software Defined Networks*, London, 2014, pp. 7-12, doi: 10.1109/EWSDN.2014.10.

[13] Andrew R. Curtis, Jeffrey C. Mogul, Jean Tourrilhes, Praveen Yalagandula, Puneet Sharma, and Sujata Banerjee. 2011. DevoFlow: scaling flow management for high-performance networks. SIGCOMM Comput. Commun. Rev. 41, 4 (August 2011), 254–265. DOI:https://doi.org/10.1145/2043164.2018466

[14] Pfaff, B., Davie, B., December 2013. The open vSwitch database management protocol, informational, internet engineering task force, RFC 7047. [Online]. Available: http://www.ietf.org/rfc/rfc7047.txt.

[15] ONF-TS-016, 2014. OpenFlow management and configuration protocol. Accessed on: 31-May-2020. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow-config/of-config-1.2.pdf

[16] E. Haleplidis *et al.*, "Network Programmability With ForCES," in *IEEE Communications Surveys & Tutorials*, vol. 17, no. 3, pp. 1423-1440, thirdquarter 2015, doi: 10.1109/COMST.2015.2439033.

[17] Smith, M., Dvorkin, M., Laribi, V., Pandey, V., Gerg, P., Weidenbacher, N., OpFlex control protocol. Accessed on: 31-May-2020. [Online]. Available: https://tools.ietf.org/html/draft-smith-opflex-03.

[18] Enns, R., Bjorklund, M., Schoenwaelder, J., Bierman, A., 2011. Network configuration protocol (netconf). Accessed on: 31-May-2020. [Online]. Available: https://www.rfceditor.org/rfc/pdfrfc/rfc6241.txt.pdf.

[19] P. L. Ventre, M. M. Tajiki, S. Salsano and C. Filsfils, "SDN Architecture and Southbound APIs for IPv6 Segment Routing Enabled Wide Area Networks," in *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1378-1392, Dec. 2018, doi: 10.1109/TNSM.2018.2876251.

[20] ONOS Wiki. [Online]. [Available]: https://wiki.onosproject.org/display/ONOS/ONOS

[21] OpenDaylight Wiki. [Online]. [Available]: https://wiki.opendaylight.org/

[22] OpenKilda Wiki. [Online]. [Available]: https://docs.open-kilda.org/xwiki/bin/view/Main/

[23] Ryu project team, "Ryu SDN framework, Release 1.0: using OpenFlow 1.3,". [Online]. [Available]: https://book.ryu-sdn.org/en/Ryubook.pdf

[24] Faucet Documentation. [Online]. [Available]: https://docs.faucet.nz/en/latest/

[25] Ericsson. Cloud SDN. Accessed on: 01-June-2020 [Online]. Available: https://www.ericsson.com/en/portfolio/digital-services/cloud-infrastructure/cloud-sdn

[26] Huawei. IMaster NCE-Fabric. Accessed on: 01-June-2020 [Online]. Available: https://e.huawei.com/es/products/network-management-and-analysis-software/imaster-nce-fabric

[27] NEC. PF6800 ProgrammableFlow Controller. Accessed on: 01-June-2020 [Online]. Available: https://www.necam.com/sdn/Software/SDNController/

[28] Open Networking Foundation (ONF), TAPI v2.1.2 Reference Implementation TR-5XX Overview Version 0.1. August 2019. [Online]. [Available]: https://wiki.opennetworking.org/display/OTCC/TAPI

[29] D. Ceccarelli and Y. Lee, “Framework for Abstraction and Control of TE Networks (ACTN),” IETF RFC 8453. August 2018.

[30] A. Mayoral et al., "Control orchestration protocol: Unified transport API for distributed cloud and network orchestration," in IEEE/OSA Journal of Optical Communications and Networking, vol. 9, no. 2, pp. A216-A222, Feb. 2017, doi: 10.1364/JOCN.9.00A216.

[31] ONOS Wiki. Projects, ACTN (Abstraction and Control of TE networks). Accessed on: 02-June-2020. Available: https://wiki.onosproject.org/pages/viewpage.action?pageId=8424694

[32] Network Functions Virtualisation (NFV); Management and Orchestration. ETSI GS NFV-MAN 001 V1.1.1, December 2014.

[33] OpenBaton, “OpenBaton - An extensible and customizable NFV MANO-compliant framework,” https://openbaton.github.io/.

[34] Cloudify “”. https://cloudify.co/

[35] Tacker, “Tacker - OpenStack NFV Orchestration platform,” https://wiki.openstack.org/wiki/Tacker.

[36] ONAP, “Open Network Automation Platform,” https://www.onap.org/.

[37] OSM, “Open Source Mano,” https://osm.etsi.org/.

[38] Intel OpenNESS https://builders.intel.com/university/networkbuilders/coursescategory/open-network-edge-services-software-openness.

[39] ONF Aether https://www.opennetworking.org/aether/

[40] 36.300, 3GPP TS, “E-UTRA and E-UTRAN; Overall description; Stage 2,” V9.10.0, Dec. 2012.

[41] 3GPP TS 23.401, “General Packet Radio Service (GPRS) enhancements for Evolved Universal Terrestrial Radio Access Network (E-UTRAN) access,” V9.16.0, Dec. 2014.

[42] 3GPP TS 23.251, “3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Network Sharing,” V9.5.0, Dec. 2012.

[43] 3GPP TR 23.707, “Architecture Enhancements for Dedicated Core Networks; Stage 2,” V13.0.0, Dec. 2014.

[44] 3GPP TR 23.711, “3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Enhancements of Dedicated Core Networks selection mechanism,” V14.0.0, Sep. 2016.

[45] 5G CLARITY Deliverable D2.2, “Primary system architecture,” Oct. 2020.

[46] 3GPP TR 28.801, “Study on management and orchestration of network slicing for next generation network (Release 15),” V15.1.0, Jan. 2018.

[47] P. Muñoz, O. Adamuz-Hinojosa, J. Navarro-Ortiz, O. Sallent and J. Pérez-Romero, “Radio Access Network Slicing Strategies at Spectrum Planning Level in 5G and Beyond,” IEEE Access, vol. 8, pp. 79604-79618, May 2020.

[48] O. Sallent, J. Pérez-Romero, R. Ferrus and R. Agusti, “On Radio Access Network Slicing from a Radio Resource Management Perspective,” IEEE Wireless Communications, vol. 24, no. 5, pp. 166-174, Oct. 2017.

[49] 5GENESIS Consortium, “5GENESIS website,” [Online]. Available: https://5genesis.eu/ . [Accessed 26 May 2020].

[50] MATILDA Consortium, “MATILDA website,” [Online]. Available: https://www.matilda-5g.eu/ . [Accessed 26 May 2020].

[51] IoRL Consortium, “IoRL website,” [Online]. Available: https://iorl.5g-ppp.eu/. [Accessed 26 May 2020].

[52] H. Alshaer and H. Haas, “Bidirectional LiFi Attocell Access Point Slicing Scheme,” IEEE Transactions on Network and Service Management, vol. 15, no. 3, pp. 909-922, Sept. 2018.

[53] J. J. Aleixendri, A. Betzler and D. Camps-Mur, “A practical approach to slicing Wi-Fi RANs in future 5G networks,” in IEEE Wireless Communications and Networking Conference (WCNC), Marrakesh, Morocco, 2019.

[54] 3GPP TR 23.793, “Study on access traffic steering, switch and splitting support in the 5G system architecture (Release 16),” V16.0.0, Dec. 2018.

[55] 5G-TOURS, “D3.1 - Baseline arquitecture and deployment objectives,” Oct. 2019.

[56] C. Casetti, C. F. Chiasserini, N. Molner, J. Martín-Pérez, T. Deiß, C.-T. Phan, F. Messaoudi, G. Landi and J. Brenes Baranzano, “Arbitration Among Vertical Services,” 2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), pp. 153-157, 2018.

[57] 5GROWTH, “D2.1 - Initial Design of 5G E2E Service Platform”.

[58] “5G-CLARITY website,” [Online]. Available: https://www.5G-CLARITY.com/. [Accessed September 2020].

[59] J. Ordonez-Lucena, P. Ameigeiras, D. Lopez, J. J. Ramos Muñoz, J. Lorca and J. Folgueira, “Network Slicing for 5G with SDN/NFV: Concepts, Architectures, and Challenges,” IEEE Communications Magazine, vol. 55, no. 5, pp. 80-87, May 2017.

[60] R. Ferrús, O. Sallent, J. Pérez-Romero and R. Agusti, “On the automation of RAN slicing provisioning: solution framework and applicability examples,” EURASIP Journal on Wireless Communications and Networking, vol. 2019, no. 1, p. 167, 2019.

[61] A. de la Oliva, X. Li, X. Costa-Perez, C. J. Bernardos, P. Bertin, P. Iovanna, T. Deiss, J. Mangues, A. Mourad, C. Casetti, J. E. Gonzalez and A. Azcorra, “5G-TRANSFORMER: Slicing and Orchestrating Transport Networks for Industry Verticals,” IEEE Communications Magazine, vol. 56, no. 8, pp. 78-84, 2018.

[62] G. Garcia-Aviles, M. Gramaglia, P. Serrano and A. Banchs, “POSENS: A Practical Open Source Solution for E2E Network Slicing,” IEEE Wireless Communications, vol. 25, no. 5, pp. 30-37, Oct. 2018.

[63] I. Afolabi, T. Taleb, P. A. Frangoudis, M. Bagaa and A. Ksentini, “Network Slicing-Based Customization of 5G Mobile Services,” IEEE Network, vol. 33, no. 5, pp. 134-141, 2019.

[64] K. Han, S. Li, S. Tang, H. Huang, S. Zhao, G. Fu and Z. Zhu, "Application-Driven E2E Slicing: When Wireless Network Virtualization Orchestrates With NFV-Based Mobile Edge Computing," IEEE Access, vol. 6, pp. 26567-26577, 2018.

[65] L. Feng, Y. Zi, W. Li, F. Zhou, P. Yu and M. Kadoch, "Dynamic Resource Allocation With RAN Slicing and Scheduling for uRLLC and eMBB Hybrid Services," IEEE Access, vol. 8, pp. 34538-34551, Feb. 2020.

[66] C. E. Rothenberg, "Fluid Network Planes – An overview of Network Refactoring and Offloading Trends," June 2019. [Online]. Available: https://netsoft2019.ieee-netsoft.org/wp-content/uploads/sites/99/2019/07/NS_Keynote3_CER.pdf. [Accessed September 2020].

[67] ETSI ENI https://www.etsi.org/technologies/experiential-networked-intelligence

[68] Apache Hadoop: "An open-source software for reliable, scalable, distributed computing" [Online] https://hadoop.apache.org/docs/current/ [Accessed July 2020].

[69] Presto: "A high performance, distributed SQL query engine for big data" [Online]. Available: https://prestosql.io/docs/current/ [Accessed July 2020].

[70] Apache Spark: "An unified analytics engine for large-scale data processing" [Online]. Available: https://spark.apache.org/docs/latest/ [Accessed July 2020].

[71] IETF RFC 1098, "A Simple Network Management Protocol (SNMP)".

[72] IETF RFC 1907, "Management Information Base (MIB) for the Simple Network Management Protocol (SNMP)"

[73] IETF RFC 6020, "YANG – A Data Modelling Language for the Network Configuration Protocol (NETCONF)"

[74] IETF RFC 4741, "Network Configuration Protocol (NETCONF)".

[75] IETF RFC 8040, "RESTCONF Protocol".

[76] "gRPC: A high-performance, open source universal RPC framework". [Online]: https://grpc.io

[77] ETSI White Paper No.31, "NGSI-LD API: for Context Information Management", 1st Edition, January 2020. [Online] Available: https://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp31_NGSI_API.pdf

[78] ETSI GS CIM 009, "Context Information Management (CIM); NGSI-LD API", v1.2.2, February 2020 [Online]. Available: https://www.etsi.org/deliver/etsi_gs/CIM/001_099/009/01.02.02_60/gs_cim009v010202p.pdf

[79] B. Claise, J. Quilbeuf, Y. El Fathi, D. Lopez and D. Voyer, "Service Assurance for Intent-Based Networking Architecture", IETF OPSAWG [Online]. Avaliable: https://tools.ietf.org/html/draft-claise-opsawg-service-assurance-architecture-02

[80] MultiPath TCP – Linux Kernel implementation, https://www.multipath-tcp.org/

[81] 3GPP TS 28.530, "5G; Management and Orchestration; Concepts, use cases and requirements"

[82] 5G-VINNI D3.1, "Specification of services delivered by each of the 5G-VINNI facilities", June 2019. [Online]: https://zenodo.org/record/3345612#.X4BkHy-w1TY

[83] J. Wang, J. Tang, Z. Xu, Y. Wang, G. Xue, X. Zhang and D. Yang, "Spatiotemporal modeling and prediction in cellular networks: A big data enabled deep learning approach," in IEEE INFOCOM, Atlanta, GA, 2017.

[84] V. Sciancalepore, K. Samdanis, X. Costa-Perez, D. Bega, M. Gramaglia and A. Banchs, "Mobile traffic forecasting for maximizing 5G network slicing resource utilization," in IEEE INFOCOM, Atlanta, GA, 2017.

[85] V. Sciancalepore, X. Costa-Perez and A. Banchs, “RL-NSB: Reinforcement Learning-Based 5G Network Slice Broker,” IEEE/ACM Transactions on Networking, vol. 27, no. 4, pp. 1543-1557, Aug 2019.

[86] Bega, Dario, Marco Gramaglia, Marco Fiore, Albert Banchs, and Xavier Costa-Perez. "DeepCog: Optimizing resource provisioning in network slicing with AI-based capacity forecasting." IEEE Journal on Selected Areas in Communications 38, no. 2 (2019): 361-376.

[87] H. Chergui and C. Verikoukis, “Offline SLA-Constrained Deep Learning for 5G Networks Reliable and Dynamic E2E Slicing,” IEEE Journal on Selected Areas in Communications, vol. 38, no. 2, pp. 350-360, Feb 2020.

[88] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk and Y. Bengio, “Learning Phrase Representations using RNN Encoder?Decoder for Statistical Machine Translation,” in Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 2014.

[89] H. Jaeger and H. Haas, “Harnessing nonlinearity: predicting chaotic systems and saving energy,” Science, vol. 304, no. 5667, pp. 78-80, 2004.

[90] A. Rodan and P. Tino, “Minimum Complexity Echo State Network,” IEEE Transactions on Neural Networks, vol. 22, no. 1, pp. 131-144, Jan. 2011.

[91] M. Lukoševičius, “A Practical Guide to Applying Echo State Networks,” in Neural Networks: Tricks of the Trade, Lecture Notes in Computer Science, Berlin, Heidelberg, Springer, 2012.

[92] 3GPP, “Study on access traffic steering, switch and splitting support in the 5G system architecture,” TR 23.793 v16.0.0, Dec. 2018.

[93] Q. Zhao, M. Chen, P. Du, T. Le and M. Gerla, “Towards Efficient Cellular Traffic Offloading via Dynamic MPTCP Path Configuration with SDN,” in International Conference on Computing, Networking and Communications (ICNC), Honolulu, HI, USA, 2019.

[94] Z. Xu, J. Tang, C. Yin, Y. Wang and G. Xue, “Experience-Driven Congestion Control: When Multi-Path TCP Meets Deep Reinforcement Learning,” IEEE Journal on Selected Areas in Communications, vol. 37, no. 6, pp. 1325-1336, Jun 2019.

[95] H. Zhang, W. Li, S. Gao, X. Wang and B. Ye, “ReLeS: A Neural Adaptive Multipath Scheduler based on Deep Reinforcement Learning,” in IEEE INFOCOM, Paris, France, 2019.

[96] IETF, [Online]. Available: https://datatracker.ietf.org/meeting/104/materials/slides-104-intarea-generic-multi-access-gma-convergence-encapsulation-protocols-00.

[97] J. Wang, C. Jiang, H. Zhang, X. Zhang, V. C. M. Leung and L. Hanzo, “Learning-Aided Network Association for Hybrid Indoor LiFi-Wi-Fi Systems,” IEEE Transactions on Vehicular Technology, vol. 67, no. 4, pp. 3561-3574, Apr 2018.

[98] Z. Du, C. Wang, Y. Sun and G. Wu, “Context-Aware Indoor VLC/RF Heterogeneous Network Selection: Reinforcement Learning With Knowledge Transfer,” IEEE Access, vol. 6, pp. 33275-33284, 2018.

[99] A. Keshavarz-Haddad, E. Aryafar, M. Wang and M. Chiang, “HetNets Selection by Clients: Convergence, Efficiency, and Practicality,” IEEE/ACM Transactions on Networking, vol. 25, no. 1, pp. 406-419, Feb 2017.

[100] M. Lanctot, V. Zambaldi, A. Gruslys, A. Lazaridou, K. Tuyls, J. Perolat, D. Silver and T. Graepel, “A Unified Game-Theoretic Approach to Multiagent Reinforcement Learning,” in Advances in Neural Information Processing Systems, 2017.

[101] A. Zappone, M. D. Renzo and M. Debbah, “Wireless Networks Design in the Era of Deep Learning: Model-Based, AI-Based, or Both?,” IEEE Transactions on Communications, vol. 67, no. 10, pp. 7331-7376, Oct 2019.

[102] [Online]. Available: https://github.com/intrig-unicamp/mininet-Wi-Fi.

[103] [Online]. Available: https://github.com/ardimasp/owcsimpy/.

[104] [Online]. Available: https://5g-lena.cttc.es/.

[105] [Online]. Available: https://github.com/mininet/mininet/wiki/Link-modeling-using-ns-3.

[106] D. Marabissi and R. Fantacci, "Highly Flexible RAN Slicing Approach to Manage Isolation, Priority, Efficiency," IEEE Access, vol. 7, pp. 97130-97142, 2019.

[107] J. Gang and V. Friderikos, "Optimal resource sharing in multi-tenant 5G networks," in IEEE Wireless Communications and Networking Conference (WCNC), Barcelona, Spain, 2018.

[108] P. Vo, M. Nguyen, T. Le and N. Tran, "Slicing the Edge: Resource Allocation for RAN Network Slicing," IEEE Wireless Communications Letters, vol. 7, no. 6, pp. 970-973, 2018.

[109] J. Perez-Romero and O. Sallent, "Optimization of Multitenant Radio Admission Control through a Semi-Markov Decision Process," IEEE Transactions on Vehicular Technology, vol. 69, no. 1, pp. 862-875, 2020.

[110] A. Alfoudi, S. Newaz, A. Otebolaku, G. Lee and R. Pereira, "An Efficient Resource Management Mechanism for Network Slicing in a LTE Network," IEEE Access, vol. 7, pp. 89441-89457, 2019.

[111] J. Pérez-Romero, O. Sallent, R. Ferrús and R. Agustí, "Profit-Based Radio Access Network Slicing for Multi-tenant 5G Networks," in European Conference on Networks and Communications (EuCNC), Valencia, Spain, 2019.

[112] Ö. U. Akgül, I. Malanchini and A. Capone, "Dynamic Resource Trading in Sliced Mobile Networks," IEEE Transactions on Network and Service Management, vol. 16, no. 1, pp. 220-233, 2019.

[113] J. Shi, H. Tian, S. Fan, P. Zhao and K. Zhao, "Hierarchical Auction and Dynamic Programming Based Resource Allocation (HA&DP-RA) Algorithm for 5G RAN Slicing," in 24th Asia-Pacific Conference on Communications (APCC), Ningbo, China, 2018.

[114] P. Caballero, A. Banchs, G. De Veciana and X. Costa-Pérez, "Network Slicing Games: Enabling Customization in Multi-Tenant Mobile Networks," IEEE/ACM Transactions on Networking, vol. 27, no. 2, pp. 662-675, 2019.

[115] J. Pérez-Romero, O. Sallent, R. Ferrús and R. Agustí, "Self-optimized admission control for multitenant radio access networks," in IEEE 28th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), Montreal, Canada, 2017.

[116] Z. Xiong, Y. Zhang, D. Niyato, R. Deng, P. Wang and L. Wang, "Deep Reinforcement Learning for Mobile 5G and Beyond: Fundamentals, Applications, and Challenges," IEEE Vehicular Technology Magazine, vol. 14, no. 2, pp. 44-52, 2019.

[117] K. Arulkumaran, M. Deisenroth, M. Brundage and A. Bharath, "Deep Reinforcement Learning: A Brief Survey," IEEE Signal Processing Magazine, vol. 34, no. 6, pp. 26-38, 2017.

[118] R. Li, Z. Zhao, Q. Sun, C.-L. I, C. Yang, X. Chen, M. Zhao and H. Zhang, "Deep Reinforcement Learning for Resource Management in Network Slicing," IEEE Access, no. 2018, pp. 74429-74441, 2018.

[119] C. Qi, Y. Hua, R. Li, Z. Zhao and H. Zhang, "Deep Reinforcement Learning with Discrete Normalized Advantage Functions for Resource Management in Network Slicing," IEEE Communications Letters, vol. 23, no. 8, pp. 1337-1341, 2019.

[120] G. Sun, Z. Gebrekidan, G. Boateng, D. Ayepah-Mensah and W. Jiang, "Dynamic Reservation and Deep Reinforcement Learning Based Autonomous Resource Slicing for Virtualized Radio Access Networks," IEEE Access, vol. 7, pp. 45758-45772, 2019.

[121] G. Sun, K. Xiong, G. Boateng, D. Ayepah-Mensah, G. Liu and W. Jiang, “Autonomous Resource Provisioning and Resource Customization for Mixed Traffics in Virtualized Radio Access Network,” IEEE Systems Journal, vol. 13, no. 3, pp. 2454-2465, 2019.

[122] Y. Abiko, T. Saito, D. Ikeda, K. Ohta, T. Mizuno and H. Mineno, “Flexible Resource Block Allocation to Multiple Slices for Radio Access Network Slicing Using Deep Reinforcement Learning,” IEEE Access, vol. 8, pp. 68183-68198, 2020.

[123] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg and D. Hassabis, “Human-level control through deep reinforcement learning,” Nature, vol. 518, pp. 529-533, 2015.

[124] S. Wang, H. Liu, P. H. Gomes and B. Krishnamachari, "Deep Reinforcement Learning for Dynamic Multichannel Access," p. 5.

[125] S. Wang, H. Liu, P. H. Gomes and B. Krishnamachari, "Deep Reinforcement Learning for Dynamic Multichannel Access in Wireless Networks’," IEEE Trans. Cogn. Commun. Netw, vol. 4, pp. 257–265,, 6 2018.

[126] Zhu, J., Song, Y., Jiang, D. and Song, H., "A New Deep-Q-Learning-Based Transmission Scheduling Mechanism for the Cognitive Internet of Things," IEEE Internet of Things Journal, pp. 2375-2385, 2017.

[127] O. Naparstek and K. Cohen, "Deep Multi-User Reinforcement Learning for Dynamic Spectrum Access in Multichannel Wireless Networks’," in GLOBECOM 2017 - 2017 IEEE Global Communications Conference, 2017.

[128] N. Zhao, Y.-C. Liang, D. Niyato, Y. Pei and Y. Jiang, "Deep Reinforcement Learning for User Association and Resource Allocation in Heterogeneous Networks’," in 2018 IEEE Global Communications Conference (GLOBECOM, 2018.

[129] C. Zhang, Z. Liu, B. Gu, K. Yamori and Y. Tanaka, "A Deep Reinforcement Learning Based Approach for Cost- and Energy-Aware Multi-Flow Mobile Data Offloading’," IEICE Trans. Commun, pp. 1625–1634,, 7 2018.

[130] J. Li, H. Gao, T. Lv and Y. Lu, "Deep reinforcement learning based computation offloading and resource allocation for MEC’," in 2018 IEEE Wireless Communications and Networking Conference (WCNC), Apr, 2018.

[131] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji and M. Bennis, "Optimized Computation Offloading Performance in Virtual Edge Computing Systems Via Deep Reinforcement Learning’," IEEE Internet Things J, vol. 6, pp. 4005–4018,, 6 2019.

[132] Ye, J. and Zhang, Y.J.A., "DRAG: Deep Reinforcement Learning Based Base Station Activation in Heterogeneous Networks," IEEE Transactions on Mobile Computing, 2019.

[133] L. Quan, Z. Wang and F. Ren, "A Novel Two-Layered Reinforcement Learning for Task Offloading with Tradeoff between Physical Machine Utilization Rate and Delay’," Future Internet, vol. 10, pp. 60,, 7 2018.

[134] Van Le, D. and Tham, C.K., "Quality of Service Aware Computation Offloading in an Ad-Hoc Mobile Cloud," IEEE Journals & Magazine, pp. 8890-8904, 2018.

[135] Tang, Zhiqing, Xiaojie Zhou, Fuming Zhang, Weijia Jia, and Wei Zhao. "Migration modeling and learning algorithms for containers in fog computing." IEEE Computer Architecture Letters 12, no. 05 (2019): 712-725.

[136] K. P. Murphy, “Machine Learning: A Probabilistic Perspective,” Massachusetts Institute of Technology, 2012.

[137] C. Bellinger, Sh. Sharma, and N. Japkowicz, "One-Class versus Binary Classification: Which and When?," in 2012 11th International Conference on Machine Learning and Applications, Boca Raton, FL, 2012.

[138] X. Li, X. Cai, Y. Hei, and R. Yuan, "nLoS identification and mitigation based on channel state information for indoor Wi-Fi localisation," in IET Communications, 2017.

[139] M. Ramadan, V. Sark, J. Gutierrez, and E. Grass "nLoS Identification for Indoor Localization using Random Forest Algorithm," in WSA 2018; 22nd International ITG Workshop on Smart Antennas, Bochum, Germany, 2018.

[140] F. Xiao, Zh. Guo, H. Zhu, X. Xie, and R. Wang, "AmpN: Real-time LoS/nLoS identification with Wi-Fi," in IEEE International Conference on Communications (ICC), Paris, 2017.

[141] S. S. Dhanjal, M. Ghaffari, and R. M. Eustice, "DeepLocNet: Deep Observation Classification and Ranging Bias Regression for Radio Positioning Systems," in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Macau, China, 2019.

[142] J. Fan and A. S. Awan, "Non-Line-of-Sight Identification Based on Unsupervised Machine Learning in Ultra Wideband Systems," IEEE Access, vol. 7, pp. 32464-32471, 2019.

[143] Zh. Xiao, H. Wen, A. Markham, N. Trigoni, P. Blunsom, and J. Frolik, "Non-Line-of-Sight Identification and Mitigation," IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, 2015.

[144] T. Guo and A. Suárez, "Enabling 5G RAN Slicing with EDF Slice Scheduling," IEEE Transactions on Vehicular Technology, vol. 68, no. 3, pp. 2865-2877, 2019.

[145] D. Marabissi and R. Fantacci, "Highly Flexible RAN Slicing Approach to Manage Isolation, Priority, Efficiency," IEEE Access, vol. 7, pp. 97130-9714, 2019.

[146] O. Al-Khatib, W. Hardjawana and B. Vucetic, "Spectrum Sharing in Multi-Tenant 5G Cellular Networks: Modeling and Planning," IEEE Access, vol. 7, pp. 1602-1616, 2019.

[147] M. Ayyash et al., "Coexistence of Wi-Fi and LiFi toward 5G: concepts, opportunities, and challenges," IEEE Communications Magazine, Vols. 64-71, no. 2, p. 54, 2016.

[148] O. Galinina, A. Pyattaev, S. Andreev, M. Dohler and Y. Koucheryavy, "5G Multi-RAT LTE-Wi-Fi Ultra-Dense Small Cells: Performance Dynamics, Architecture, and Trends," IEEE Journal on Selected Areas in Communications, vol. 33, no. 6, pp. 1224-1240, 2015.

[149] K.Koutlia, A. Umbert, R. Riggio, I. Vilà and F. Casadevall, "A new RAN slicing strategy for multi-tenancy support in a WLAN scenario," in 4th IEEE Conference on Network Softwarization and Workshops (NetSoft), Montreal, QC, Canada, 2018.

[150] N. Makris, C. Zarafetas, P. Basaras, T. Korakis, N. Nikaein and L. Tassiulas, "Cloud-Based Convergence of Heterogeneous RANs in 5G Disaggregated Architectures," in IEEE International Conference on Communications (ICC), Kansas City, MO, 2018.

[151] H. Koumaras et al., "5GENESIS: The Genesis of a flexible 5G Facility," in IEEE 23rd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), Barcelona, 2018.

[152] D. Camps-Mur et al., "5G-PICTURE: A Programmable Multi-tenant 5G Compute-RAN-Transport Infrastructure," in European Conference on Networks and Communications (EuCNC), Valencia, Spain, 2019.

[153] J. Ha and Y. Choi, "Support of a Multi-access Session in 5G Mobile Network," in 25th Asia-Pacific Conference on Communications (APCC), Ho Chi Minh City, Vietnam, 2019.

[154] D. M. Gutierrez-Estevez G.-E. et al., "Artificial Intelligence for Elastic Management and Orchestration of 5G Networks," IEEE Wireless Communications, vol. 26, no. 5, pp. 134-141, 2019.

[155] V. Sciancalepore, X. Costa-Perez and A. Banchs, "RL-NSB: Reinforcement Learning-Based 5G Network Slice Broker," IEEE/ACM Transactions on Networking, vol. 27, no. 4, pp. 1543-1557, 2019.

[156] H. Xiang, S. Yan and M. Peng, "A Realization of Fog-RAN Slicing via Deep Reinforcement Learning," IEEE Transactions on Wireless Communications, vol. 19, no. 4, pp. 2515-2527, 2020.

[157] G. Sun, Z. T. Gebrekidan, G. O. Boateng, D. Ayepah-Mensah and W. Jiang, "Dynamic Reservation and Deep Reinforcement Learning Based Autonomous Resource Slicing for Virtualized Radio Access Networks," IEEE Access, vol. 7, pp. 45758-45772, 2019.

[158] X. Chen et al., "Multi-Tenant Cross-Slice Resource Orchestration: A Deep Reinforcement Learning Approach," IEEE Journal on Selected Areas in Communications, vol. 37, no. 10, pp. 2377-2392, 2019.

[159] R. Li et al., "Deep Reinforcement Learning for Resource Management in Network Slicing," IEEE Access, vol. 6, pp. 74429-74441, 2018.

[160] A. Thantharate, R. Paropkari, V. Walunj and C. Beard, "DeepSlice: A Deep Learning Approach towards an Efficient and Reliable Network Slicing in 5G Networks," in IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), New York City, USA, Oct. 2019.

[161] B. Han, J. Lianghai and H. D. Schotten, "Slice as an Evolutionary Service: Genetic Optimization for Inter-Slice Resource Management in 5G Networks," IEEE Access, vol. 6, pp. 33137-33147, 2018.

[162] https://www.itu.int/en/ITU-T/focusgroups/ml5g/Pages/default.aspx.

[163] https://www.etsi.org/committee/1423-eni.

[164] ITU-T, "Machine learning in future networks including IMT-2020: Use cases," Y.3170-series Supplement 55, October 2019.

[165] O.-R. Alliance, "O-RAN: Towards an Open and Smart RAN," White Paper, Oct. 2018.

[166] 3GPP, "Study on access traffic steering, switch and splitting support in the 5G system architecture," TR 23.793 v16.0.0, Dec. 2018.

[167] [Online]. Available: https://github.com/keras-rl/keras-rl.

[168] [Online]. Available: https://github.com/tensorforce/tensorforce.

[169] T. Le Duc, R. García Leiva, P. Casari and P.-O. Östberg, "Machine Learning Methods for Reliable Resource Provisioning in Edge-Cloud Computing: A Survey," *ACM Comput. Surv.,* vol. 52, no. 5, p. 39, 2019.

[170] I. Afolabi, J. Prados-Garzon, M. Bagaa, T. Taleb and P. Ameigeiras, "Dynamic Resource Provisioning of a Scalable E2E Network Slicing Orchestration System," *IEEE Transactions on Mobile Computing,* vol. 19, no. 11, pp. 2594 - 2608, 2020.

[171] R. Moreno-Vozmediano, R. S. Montero, E. Huedo and I. M. Llorente, "Efficient resource provisioning for elastic Cloud services based on machine learning techniques," *J Cloud Comp,* vol. 8, no. 5, 2019.

[172] C. H. T. Arteaga, F. Rissoi and O. M. C. Rendon, "An Adaptive Scaling Mechanism for Managing Performance Variations in Network Functions Virtualization: A Case Study in an NFV-based EPC," in *13th International Conference on Network and Service Management (CNSM)*, Tokyo, Japan, Nov. 2017.

[173] R. Mijumbi, S. Hasija, S. Davy, A. Davy, B. Jennings and R. Boutaba, "Topology-Aware Prediction of Virtual Network Function Resource Re-quirements," *IEEE Transactions on Network and Service Management,* vol. 14, no. 1, p. 106–120, 2017.

[174] N. Navet, T. L. Mai and J. Migge, "Using machine learning to speed up the design space exploration of ethernet tsn networks," Tech. Rep., University of Luxembourg, 2019.

[175] J. Prados-Garzon, T. Taleb and M. Bagaa, "LEARNET: Reinforcement Learning Based Flow Scheduling for Asynchronous Deterministic Networks," in *IEEE Int. Conf. on Commun. (ICC)*, Virtual Conference, June 2020.

[176] RAN Alliance, O-RAN Use Cases and Deployment Scenarios, Towards Open and Smart RAN, White Paper, February 2020

[177] "Amazon SageMaker," [Online]. Available: https://aws.amazon.com/sagemaker/.

[178] "Azure Machine Learning," [Online]. Available: https://docs.microsoft.com/en-us/azure/machine-learning.

[179] "DataRobot," [Online]. Available: https://www.datarobot.com/.

[180] "Valohai," [Online]. Available: https://valohai.com/.

[181] "Acumos," [Online]. Available: https://www.acumos.org/.

[182] "Acumos Overview, with SDN & ONAP support," [Online]. Available: https://marketplace.acumos.org/#/home.

[183] "H2O," [Online]. Available: https://www.h2o.ai/.

[184] "H2O Driverless AI," [Online]. Available: https://www.h2o.ai/try-driverless-ai/.

[185] "Clipper," [Online]. Available: http://clipper.ai/.

[186] "TensorFlow Serving," [Online]. Available: https://www.tensorflow.org/tfx/guide/serving.

[187] "MLflow," [Online]. Available: https://mlflow.org/.

[188] "Kubeflow," [Online]. Available: https://www.kubeflow.org/.

[189] "Apache Spark," [Online]. Available: https://spark.apache.org/.

[190] "Apache Spark MLlib," [Online]. Available: https://spark.apache.org/mllib/.

[191] E. Salova, G. Evans, "AutoML + Pentaho + Grafana for fast solution prototyping", [Online]. Available: https://towardsdatascience.com/automl-pentaho-grafana-for-fast-solution-prototyping-e3c83b7209

[192] ONF, Intent NBI – Definition, Open Networking Foundation, 2016.

[193] M. Cohen, The Intent for Intent: Notes from the Recent Intent-Based Summit, SDx Central, 2015.

[194] D. Lenrow, Intent: Don't Tell Me What to Do! (Tell Me What You Want), SDx Central, 2015.

[195] A. Lerner, Intent-based Networking, Gartner, Gartner Blog Network, 2017.

[196] OpenStack, "GroupBasedPolicy," [Online]. Available: https://wiki.openstack.org/wiki/GroupBasedPolicy. [Accessed 28 September 2020].

[197] D. Brainbridge, "Network Intent Composition in OpenDaylight," 19 May 2015. [Online]. Available: https://www.slideshare.net/opendaylight/whats-the-intent. [Accessed 28 September 2020].

[198] ONOS, "Intent Framework," [Online]. Available: https://wiki.onosproject.org/display/ONOS/Intent+Framework. [Accessed 28 September 2020].

[199] Li, C., Cheng, Y., Strassner, J., Havel, O., Liu, W., Martinez-Julia, P., Nobre, J., Lopez, D., "Intent Classification," 9 July 2019. [Online]. Available: https://www.ietf.org/archive/id/draft-li-nmrg-intent-classification-01.txt. [Accessed 28 September 2020].

[200] A. Clemm, C. L., L. Granville and J. Tantsura, Intent-Based Networking - Concepts and Definitions, IETF, Network Working Group, 2020.

[201] T. F. Z. P. Team, “5G Slicing and Management in the TM Forum,” 16 April 2018. [Online]. Available: https://wiki.onap.org/download/attachments/28382488/5G%20slicing%20and%20management%20-%20TMF%20contribution%20cleaned.pdf?api=v2. [Accessed 28 September 2020].

[202] 3GPP, Telecommunication management; Study on scenarios for Intent driven management services for mobile networks, 3GPP, SA5.

[203] F. Callegati, W. Cerroni and C. Contoli, Intent-based Network Programmability, ICIN 2019, 2019.

[204] G. Davoli, W. Cerroni, S. Tomovic, C. Buratti, C. Contoli and F. Callegati, “Intent-based service management for heterogeneous software-defined infrastructure domains,” International Journal for Network Management, vol. 29, no. 1, 2018.

[205] F. Esposit, J. Wan, C. Contoli, G. Davoli, W. Cerroni and F. Callegati, “A Behavior-Driven Approach to Intent Specification for Software-Defined Infrastructure Management,” in 2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), 2018.

[206] S. Arezoumand, K. Dzeparoska, H. Bannazadeh and A. Leon-Garcia, “MD-IDN: Multi-Domain Intent-Driven Networking in Software-Defined Infrastructures,” in 13th International Conference on Network and Service Management (CNSM), Tokyo, 2017.

[207] O.-R. Alliance, “O-RAN: Towards an Open and Smart RAN,” O-RAN Alliance, 2018.

[208] O. P. Framework, “A short Introduction to APEX,” ONAP, 21 August 2019. [Online]. Available: https://docs.onap.org/en/dublin/submodules/policy/apex-pdp.git/docs/APEX-Introduction.html. [Accessed 28 September 2020].

[209] J. F. Allen, “Maintaining knowledge about temporal intervals,” Communications of the ACM, vol. 26, no. 11, p. 832–843, 1983.

[210] Schaller, Sibylle, and Dave Hood. "Software defined networking architecture standardization." Computer standards & interfaces 54 (2017): 197-202.