H2020 5Growth Project
Grant No. 856709

D2.3: Final Design and Evaluation of the innovations of the 5G End-to-End Service Platform

Abstract

This deliverable presents in detail the 5Growth end-to-end service platform along with all the innovations to support the project field trials, including the specification of all functional blocks and interfaces, and an exhaustive performance validation of individual components.

Document properties

Document number	D2.3
Document title	Final Design and Evaluation of the Innovations of 5G End-to-End Service Platform
Document responsible	Josep Mangues-Bafalluy (CTTC)
Document editor	Josep Mangues-Bafalluy (CTTC)
Authors	Chia-Yu Chang (NBL), Andres Garcia Saavedra (NEC), Ricardo Martínez (CTTC), Jorge Baranda (CTTC), Luca Vettori (CTTC), Engin Zeydan (CTTC), Josep Mangues-Bafalluy (CTTC), Claudio Casetti (POLITO), Carla Fabiana Chiasserini (POLITO), Giuseppe Avino (POLITO), Marco Malinverno (POLITO), Corrado Puligheddu (POLITO), Giada Landi (NXW), Juan Brenes (NXW), Xi Li (NEC), Josep Xavier Salvat (NEC), Jorge Martín Pérez (UC3M), Carlos Guimarães (UC3M), Kiril Antevski (UC3M), Antonio de la Oliva (UC3M), Iván Vidal (UC3M), Agustín Caparrós (TELCA), Teresa Giner (TELCA), Manuel A. Jiménez (TELCA), Ramón Pérez (TELCA), Javier Sacido (TELCA), Aitor Zabala (TELCA), Danny De Vleeschauwer (NBL), Daniel Corujo (ITAv), David Santos (ITAv), Diogo Gomes (ITAv), João Fonseca (ITAv), João Alegria (ITAv), João Paulo Barraca (ITAv), Rafael Simões (ITAv), Vitor Cunha (ITAv), Sokratis Barmpounakis (NKUA), Lina Magoula (NKUA), Nikolaos Koursioumpas (NKUA), Nikolaos Maroulis (NKUA), Ioannis Stavrakakis (NKUA), Oleksii Kolodiazhnyi (MIRANTIS), Konstantin Tomakh (MIRANTIS), Denys Kucherenko (MIRANTIS), Fabio Ubaldi (TEI), Andrea Boddi (TEI), Giulio Bottari (TEI), Paola Iovanna (TEI), Luca Valcarenghi (SSSA), Andrea Sgambelluri (SSSA), Molka Gharbaoui (SSSA), Barbara Martini (CNIT), Annalina Ruscelli (SSSA), Piero Castoldi (SSSA), Diego R. López (TID), Carlos Marques (ALB), José Bonnet (ALB), Annachiara Pagano (TIM), Pietro Scalzo (TIM)
Target dissemination level	PU
Status of the document	Final
Version	1.0
Delivery date	May 31, 2021
Actual delivery date	May 31, 2021

Production properties

Reviewers	Ramon Pérez (TELCA), Diego R. López (TID), Andres Garcia Saavedra (NEC), Xi Li (NEC), Carlos Guimarães (UC3M)

Disclaimer

This document has been produced in the context of the 5Growth Project. The research leading to these results has received funding from the European Community's H2020 Programme under grant agreement Nº H2020-856709.

Contents

List of Figures

List of Tables

List of Acronyms

5Gr-AIMLP – 5Growth Artificial Intelligence / Machine Learning Platform

5Gr-FFB – 5Growth Forecasting Functional Block

5Gr-VoMS –5Growth Vertical-oriented Monitoring System

5Gr-RL – 5Growth Resource Layer

5Gr-SO – 5Growth Service Orchestrator

5Gr-VS – 5Growth Vertical Slicer

5GT – 5G-TRANSFORMER

5GT-MTP – 5G-TRANSFORMER Mobile Transport and Computing Platform

5GT-SO – 5G-TRANSFORMER Service Orchestration

5GT-MP – 5G-TRANSFORMER Monitoring Platform

AD – Administrative domain

ADT – Anomaly Detection

AI – Artificial Intelligence

ARIMA - Autoregressive Integrated Moving Average

BSS – Business Support System

CAPEX – Capital Expenditures

CI/CD – Continuous Integration / Continuous Delivery

CP – Control Plane

CSA – Communication Service Abstraction

CSMF – Communication Service Management Function

CU – Cloud/Central Unit

DB – Database

DLT – Distributed Leger Technology

DPM – Dynamic Profiling Mechanism

DT – Digital Twin

DU – Distributed Unit

E2E – End-to-End

EBI – Eastbound Interface

eMBB – Enhanced Mobile Broadband

ETS – Exponential Smoothing

gNB – next Generation Node B

GUI – Graphical User Interface

IDS – Intrusion Detection System

IE – Information Element

IL – Instantiation Level

InA – Infrastructure Abstraction

IPS – Intrusion Prevention System

IWL – Inter Working Layer

KPI – Key Performance Indicator

LCM – Lifecycle Management

LSTM – Long term Short Term Memory

MANO – Management and Orchestration

MEC – Multi-access Edge Computing

MF – Management Function

ML – Machine Learning

MLP – Multi-Layer Perceptron

mMTC – Massive Machine Type Communications

MILP – Mixed Integer Linear Program

MQ – Message Queue

MTD – Moving Target Defense

NBI – Northbound Interface

NFV – Network Function Virtualization

NFVI – NFV Infrastructure

NFV-NS – NFV Network Service

NFVO – NFV Orchestrator

NS – Network Slice

NSD – Network Service Descriptor

NSF – Network Service Federation

NSI – Network Slice Instance

NSMF – Network Slice Management Function

NSO – Network Service Orchestrator

NSSMF – Network Slice Subnet Management Function

NST – Network Slice Template

OPEX – Operational Expenditures

OSS – Operations Support System

OvS – Open vSwitch

PA – Placement Algorithm

PDU – Physical Device Unit

PNF – Physical Network Function

PoC – Proof of Concept

PoP – Point of Presence

QoS – Quality of Service

RAN – Radio Access Network

REST – Representational State Transfer

RIC – Radio Intelligent Controller

RO – Resource Orchestrator

(R)RU – (Remote) Radio Unit

RT – Real Time

RVM – Remote Virtual Machine

SC – Smart Contract

SBI – Southbound Interface

SDN – Software Defined Networking

SFC – Service Function Chaining

SLA – Service Level Agreement

SLPOC – Single Logical Point of Contact

SO – Service Orchestrator

TSDB – Time Series Database

UC – Use Case

UE –User Equipment

URLLC – Ultra Reliable Low Latency Communications

V2N – Vehicle to Network

VA – Vertical Application

VIM – Virtual Infrastructure Manager

VM – Virtual Machine

VNF – Virtual Network Function

VNFD – VNF Descriptor

VNFFG – VNF Forwarding Graph

VNFM – VNF Manager

vRAN – Virtual Radio Access Network

vRAP – Virtualized Radio Access Point

VSB – Vertical Service Blueprint

VSD – Vertical Service Descriptor

VSI – Vertical Slice Instance

WBI – Westbound Interface

WAN – Wide Area Network

WIM – WAN Infrastructure Manager

Executive Summary and Key Contributions

This document (together with D2.4 [9]) presents the final results of WP2. More specifically, this one focuses on the final design of the architecture and developed innovations and D2.4 on the software release of the concepts developed here. As such, it fully develops the initial architectural ideas presented in D2.1 [1]. These ideas cover the gaps that we have identified in the baseline 5G-TRANSFORMER architecture in the form of twelve innovations that were integrated into the architecture developed in this document. These innovations have been designed bearing in mind the needs of the vertical pilot use cases under study in this project. Therefore, though a continuous interaction was in place throughout the project, this document can be considered as the final outcome from WP2 to provide 5Growth´s vanilla platform for the pilots in WP3 and WP4. The final implementation of the 5Growth platform that develops these architectural concepts is described in the D2.4 companion deliverable [9] and available for download from the 5Growth repository [2].

The key contributions of D2.3, and of WP2 as a whole, are as follows:

- Description of the final 5Growth architecture (Section 2), which integrates all its building blocks, namely 5Growth Vertical Slicer (5Gr-VS), Service Orchestrator (5Gr-SO), Resource Layer (5Gr-RL), Vertical-oriented Monitoring Systems (5Gr-VoMS), AI/ML platform (5Gr-AIMLP), and Forecasting Fucntional Block (5GR-FFB)
- The rationale and final design of the twelve innovations identified in D2.1 are also presented (Section 3) along with the results obtained from their evaluations either on their own or grouped together with other innovations (Section 4):
  - Innovation 1 (I1): Support of Verticals/RAN segments in network slices;
  - Innovation 2 (I2): Support of Verticals/Vertical-service monitoring;
  - Innovation 3 (I3): Monitoring orchestration;
  - Innovation 4 (I4): Control and Management/Control-loops stability;
  - Innovation 5 (I5): Control and Management/AI/ML Support;
  - Innovation 6 (I6): End-to-End orchestration/Federation and Inter-domain;
  - Innovation 7 (I7): End-to-End orchestration/Next Generation RAN;
  - Innovation 8 (I8): Smart orchestration and resource control algorithms;
  - Innovation 9 (I9): Anomaly detection algorithms;
  - Innovation 10 (I10): Forecasting and inference;
  - Innovation 11 (I11): Security and auditability;
  - Innovation 12 (I12): 5Growth Continuous Integration and continuous delivery.

In addition to introducing the design of these innovations, this document also presents evaluation results, via simulations or small-to-medium-scale experimental prototypes. These results validate the ability of these innovations to serve their purpose. Among these, it is worth highlighting the following key featrures of the 5Growth platform:

- 5Growth can instantiate end-to-end network slices well within minutes (see Section 4.1);
- 5Growth can swiftly manage AI/ML models as a service to other components of the stack to aid in network service scaling operations efficiently (see Section 4.3);

- 5Growth can inter-connect with other domains, such as 5G-EVE (Section 4.4) and 5G-VINNI (see Section 4.5);
- 5Growth can attain substantial OPEX/CAPEX savings in virtualized RANs (see Section 4.8);
- 5Growth can autonomously scale services and provide resource arbitration across slices efficiently (see Section 4.9.1);
- 5Growth can attain performance isolation across network slices, and provide slices with performance guarantees in network bandwidth and delay (see Section 4.9.2);
- 5Growth provides novel security mechanisms that can protect all 5Growth interfaces (see Section 4.12).

These results, among others properly reported in Section 4, validate that WP2 has contributed to meet the 5GPPP and WP2 KPIs, as summarized in Table 1.

TABLE 1: WP2 KPIS

WP2 KPI	Innovations
OPEX savings	I2, I3, I4, I5, I6, I7, I8, I10
Security and reliability	I2, I3, I9, I11
Service creation time in < 90 min	I1, I6, I7, I8
Dense deployments	I1, I7, I8, I9, I10

Finally, the work reported in this deliverable has fostered a number of exploitation, communication and dissemination achievements as reported in D5.4 [103]. For instance,

- Industrial partners are moving their innovations (e.g., performance isolation, next-generation RANs) into their Business Unit product development plans and there are multiple internal ongoing efforts in this direction (e.g., exploitation-oriented workshops);
- 2 patent applications have been submitted (and more are under preparation);
- The 5Growth code has been published as open source and there have been multiple contributions to large open-source projects have been made (e.g. in networkx graph library, in ONOS, or in OSM);
- Some innovations have been showcased in relevant events/PoCs (e.g., Mobicom, Infocom, ETSI ENI PoC, OSM Ecosystem Day).

Furthermore, our innovations have been published in major scientific journals (e.g., IEEE JSAC, IEEE TMC, IEEE TWC) and conferences (e.g., ACM Mobicom, ACM CoNEXT, IEEE INFOCOM).

1 Introduction

WP2´s main goal is to design and build a 5G End-to-End Service Platform[1] that is able to cope with the demanding requirements of a set of vertical use cases in the area of industry 4.0/smart factories, smart transportation, and energy/smart grid. To this end, our first task in WP2 was to analyse the technical and functional gaps in the 5G-TRANSFORMER platform to provide some of the features that either enable 5Growth use cases or maximize their efficiency.

Our analysis, presented in D2.1 [1], gave light to twelve innovations across all functional components of the platform and two new functional components (to manage AI/ML models and forecasted metrics). Most of these innovations have also been integrated into 5Growth´s vanilla platform, which is available for WP3 and WP4 for customization and hence for implementation of the different pilot use cases described in the project, and also for the whole community through our public repositories [2]. The details of the final 5Growth software release are presented in D2.4 [9].

In summary, the innovations that WP2 has designed are the following:

- **Innovation 1 (I1): Support of Verticals/RAN segments in network slices**
  Through this innovation, 5Growth integrates 3GPP slice definitions to support slices that include the radio access network (RAN) segment, hence enabling end-to-end network slicing.
- **Innovation 2 (I2): Support of Verticals/Vertical-service monitoring.**
  This innovation enhances 5G-TRANSFORMER´s monitoring platform with a number of features that improve vertical service monitoring, including log tracking, virtual function statistics, a message brokering system, and more.
- **Innovation 3 (I3): Monitoring orchestration.**
  This innovation enables the management of vertical-oriented monitoring agents required to monitor deployed services.
- **Innovation 4 (I4): Control and Management/Control-loops stability.**
  This innovation designs a series of workflows for the integration of relevant 5Growth entities (monitoring, AI/ML platform, management and orchestration stack) to dynamically adapt to service and scenario conditions to fulfil SLA requirements.
- **Innovation 5 (I5): Control and Management/AI/ML Support.**
  This innovation has created a new functional block from scratch, named 5Growth AI/ML platform (5Gr-AIMLP). This module is intended to manage the lifecycle of models that other 5Growth components (5Gr-SO, 5Gr-VS, 5Gr-RL, 5Gr-FFB) can then use to make informed decisions.

[1] Architecture, platform and stack are used frequently throughout this document. The first term refers to the conceptual design of the building blocks developed in the project and their relationships. The other two are often used interchangeably to refer to the architecture once deployed as an operational system.

- **Innovation 6 (I6): End-to-End orchestration/Federation and Inter-domain.**
  This innovation enhances 5G-TRANSFORMER´s federation and inter-domain support to enable many of our pilot use cases (e.g., support of intergration of ICT-17 platforms).
- **Innovation 7 (I7): End-to-End orchestration/Next Generation RAN.**
  This innovation integrates O-RAN building blocks and interfaces into 5Growth stack to support next-generation RANs and provide finer-grained (virtualized) RAN control.
- **Innovation 8 (I8): Smart orchestration and resource control algorithms.**
  This innovation provides a number of algorithmic solutions to perform (i) orchestration tasks, such as function placement, service scaling or slice sharing; (ii) resource control tasks, such as resource abstraction, resource allocation or performance isolation; (iii) and traffic profiling tasks.
- **Innovation 9 (I9): Anomaly detection algorithms.**
  This innovation provides autonomous anomaly detection features to the 5Growth stack, targeting edge/RAN operation as part of slice management.
- **Innovation 10 (I10): Forecasting and inference.**
  This innovation provides a new functional block in charge of providing forecasting metrics as a service to other building blocks in 5Growth.
- **Innovation 11 (I11): Security and auditability.**
  This innovation provides a Moving Target Defence (MTD)-based scheme to protect the interaction between 5Growth management entities, and hence the whole network operation.
- **Innovation 12 (I12): 5Growth Continuous Integration and Continuous Delivery.**
  Finally, this innovation provides CI/CD workflows to enable an agile development of 5Growth components that should ease the platform customization and integration required in WP3 to implement the pilot use cases devised in the project.

As for the remaining of this document, Section 2 presents the 5Growth architecture, providing details of each of its building blocks. Section 3 describes the innovations that have been designed to cover the identified gaps. Section 4 presents the evaluations carried out of the innovations explained in Section 3. Finally, Section 5 closes the document with some concluding remarks.

2 5Growth Architecture

D2.1 [1] complemented with the information available in D2.2 [3], describes the general 5Growth architecture, which enables the automated deployment and operation of 5G customized slices to accommodate diverse targeted 5Growth verticals, such as Industry 4.0, Transportation and Energy. In a nutshell, the architecture is constituted by six key building blocks along with defined interfaces among them. These building blocks are: the Vertical Slicer (5Gr-VS), the Service Orchestrator (5Gr-SO), the Resource Layer (5Gr-RL), the Vertical-Oriented Monitoring System (5Gr-VoMS), the AI/ML Platform (5Gr-AIMLP), and the Forecasting Functional Block (5Gr-FFB). Such a baseline architecture has been continuously revisited and refined to conveniently embrace the different architecture, algorithm and framework innovations explored and investigated within WP2. These innovations indeed cover heterogeneous aspects, such as vertical service monitoring and monitoring orchestration, control-loop based operability, AI/ML support, federation and multi-domain scenarios, smart orchestration, anomaly detection and forecasting as well as security aspects.

In the following, we briefly overview the key building blocks of the 5Growth architecture (depicted in Figure 1) further detailed in the respective subsections.

- The Vertical Slicer (5Gr-VS) is the front-end point for verticals demanding the provisioning and management of vertical services via a simplified northbound interface with the vertical OSS/BSS (Operations Support System/Business Support System).
- The Service Orchestrator (5Gr-SO) handles both the network service and resource orchestration capabilities to: (i) enable end-to-end orchestration of NFV-Network Services (NFV-NS) by placing them either on a single or across multiple administrative domains (ADs) based on service requirements and availability of the services/resources offered by each of the domains; and (ii) manage their lifecycles (including on-boarding, instantiation, update, scaling, termination).

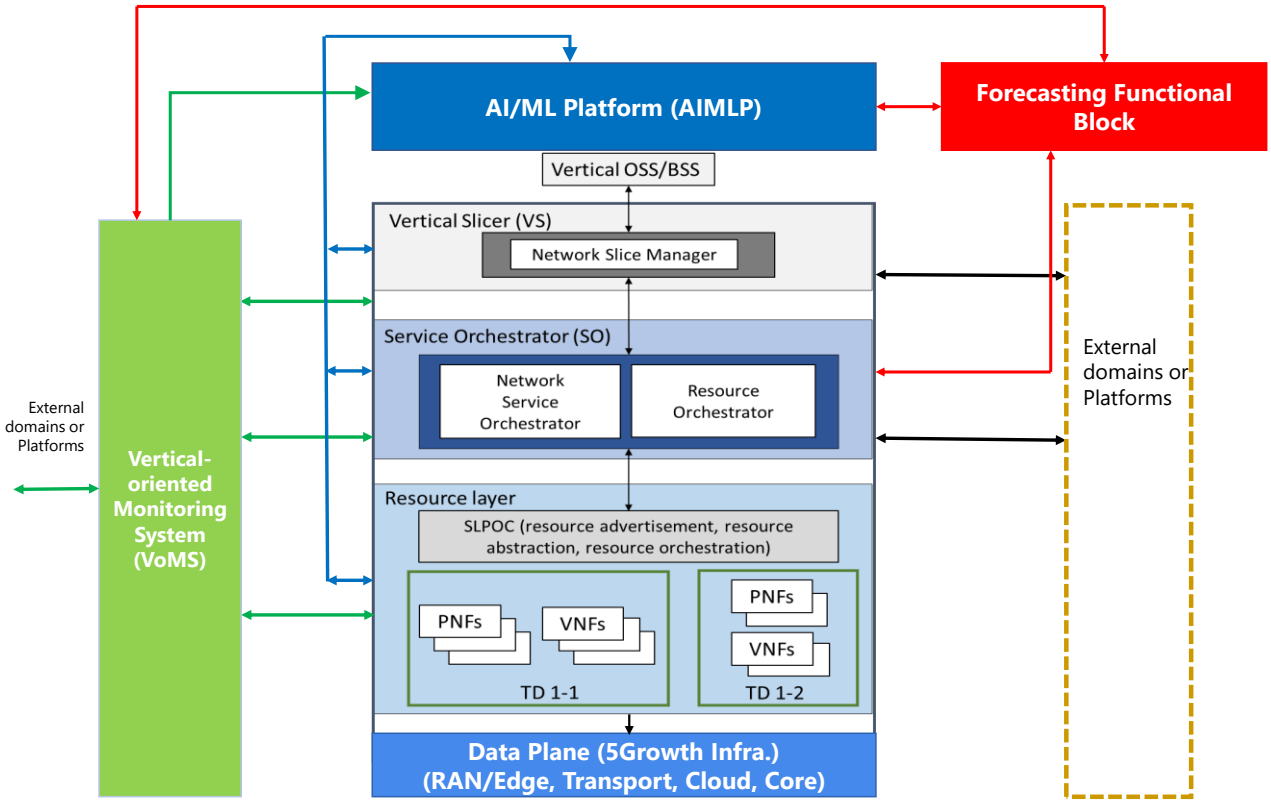


FIGURE 1: 5GROWTH HIGH-LEVEL ARCHITECTURE

- The Resource Layer (5Gr-RL) is responsible for managing the local infrastructure interacting with the underlying controllers (i.e., VIMs and WIMs). To this end, the 5Gr-RL keeps track of heterogeneous resources exposed by these controllers, such as compute, storage, and transport resources. This grants the 5Gr-RL the capability to perform resource orchestration operations over its controlled (administrative) domain. Consequently, the 5Gr-RL integrates the management of all the compute, storage, and networking (physical and virtual) resources where the elements forming both network slices and end-to-end services are accommodated.
- The Vertical-oriented Monitoring System (5Gr-VoMS) constitutes the monitoring platform which supports heterogeneous sets of services and technological domains. To do that, it offers the required functionalities, such as log aggregation, a scalable data distribution system and dynamic probe reconfiguration. The design of the 5Gr-VoMS supports scalable data distribution system to enable dynamic reconfiguration of the monitoring probes.
- The AI/ML Platform (5Gr-AIMLP) is the entity that manages (i.e., deploys, configures and trains) AI models relying on the gathered monitoring data information from the 5Gr-VoMS. This entails exposing a catalogue of AI models susceptible to be tuned/chained to come up with more complex models. Additionally, the 5Gr-AIMLP, triggered by the 5Growth-related layer (e.g., 5Gr-VS, 5Gr-SO and 5Gr-RL) retrieves on demand context information and performance metrics to complement its available datasets to update a model or compute its reward. Finally, the specific 5Growth-related layer can configure all needed data pipeline components to run the optimized model retrieved from the 5Gr-AIMLP to make the decision that allow achieving a specific targeted performance functionality / objective.

- The Forecasting Functional Block (5Gr-FFB) aims at generating forecast values of parameters monitored by the 5Gr-VoMS. These forecast values can be utilized by decision modules of the platform to take a proactive approach in reacting to changing states. The 5Gr-FFB interacts with the 5Gr-AIMLP, the 5Gr-VoMS, and other main functional elements of the 5Growth architecture, such as the 5Gr-SO. By interacting with the 5Gr-AIMLP, the 5Gr-FFB might request trained ML models to be expoited into its own forecasting tasks. By interacting with the 5Gr-VoMS, the 5G-FFB retrieves the current values of the parameters to be forecast and input the forecast data to be made available through the 5Gr-VoMS to the other decisional elements, such as the 5Gr-SO. The interaction with the 5Growth architecture decisional elements allows exchanging the information about which parameters must be forecast and how they are identified.

2.1 Vertical Slicer

The 5Growth Vertical Slicer (5Gr-VS), which evolves from the 5G-TRANSFORMER Vertical Slicer, is located at the uppermost layer of the 5Growth architecture, and acts as the entry point for the vertical industries that constitute the consumers of the 5Growth platform services. The 5Gr-VS processes the requests for vertical services instances (VSI), issued by the verticals, and is responsible for the translation and decomposition of these requests into vertical sub-services and network slice instances (VSSIs and NSIs respectively). Both VSSIs and NSIs are created on demand based on the outcome of the translation and decomposition process. Eventually, the 5Gr-VS requests the deployment of the NFV network services (NFV-NSs) required by the vertical service mediated by the 5Growth Service Orchestrator (5Gr-SO) (see next section for details).

The evolution of the 5Gr-VS mainly targeted the following innovations (we refer to the specific innovation sections for further details):

- Innovation 1 and 7: Updating the information models and the interfaces in order to enable the allocation of RAN resource as part of the network slices associated with a vertical service (see Section 3.1.1).
- Innovation 6: Updating the information models and workflows in order to allow service provisioning across multiple administrative domains (using vertical (sub)service and network slice level service decomposition). See Section 3.1.6 for further details.
- Innovation 5: Enhancing the mechanisms used to map vertical services into network slices, and enabling network slice instance sharing using smart algorithms leveraging AI/ML technologies (see Section 3.1.5).

The northbound interface (NBI) of the 5Gr-VS provides a vertical-oriented and simplified interface, which enables verticals to customize, deploy and manage vertical services using high-level primitives. To this end, the 5Gr-VS offers a catalogue of vertical service blueprints (VSBs) acting as templates for the vertical to define its service. These templates mainly model the different service components and their interconnection. The vertical service definition is then realized with the creation of a Vertical Service Descriptor (VSD), which hosts the specific values for the service-level input parameters

defined in the VSB. All the vertical service instance deployment and scaling operations are performed based on these latter descriptors, and the specific values provided in them are used as part of the vertical service translation and decomposition process. In 5Growth, VSBs and VSDs have been enhanced to convey mobile traffic information that enables the 5Growth platform to embrace the radio access network (RAN) as part of the end-to-end network slice as described in Section 3.1.1.

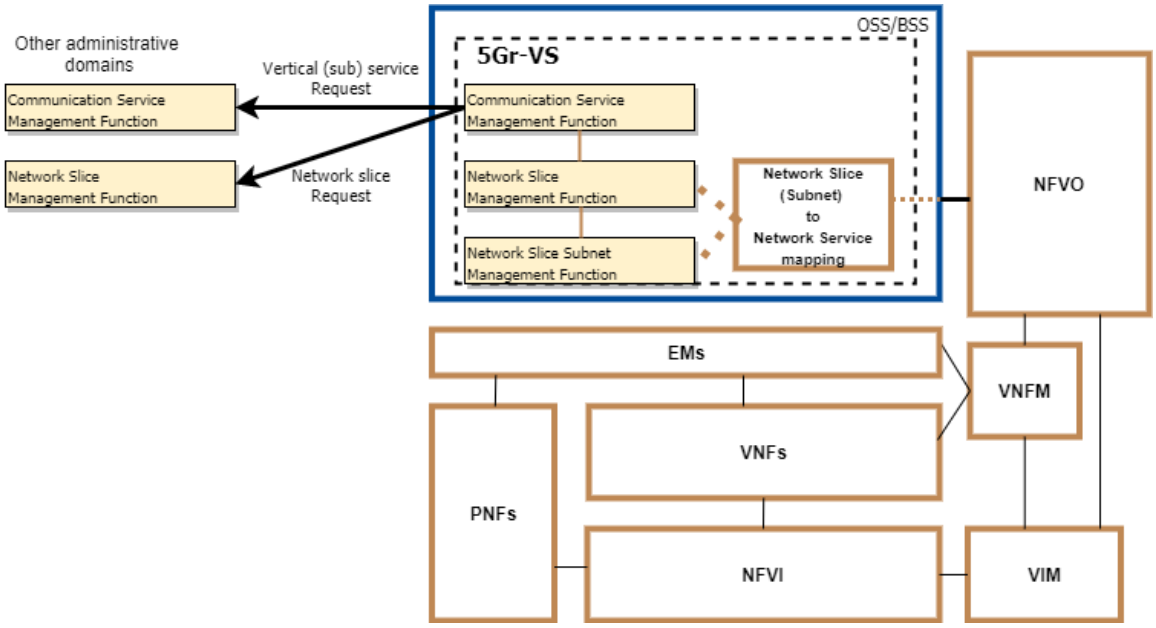


FIGURE 2: MAPPING BETWEEN 5GR-VS AND 3GPP NETWORK SLICING ARCHITECTURE

Figure 2 shows the high-level functional block of the 5Gr-VS in terms of the network slicing architecture defined in [4]. As shown in the picture, the 5Gr-VS integrates the Communication Service, Network Slice and Network Slice Subnet Management Functions (i.e., CSMF, NSMF and NSSMF, respectively). In this architecture, the CSMF is responsible for determining the network slice(s) and the specific dimension required to provision the vertical service. In the 5Gr-VS, this process is performed using translation rules that establish the network slices and the dimension required for a range of values of the VSB input parameters. In particular, a translation rule determines the Network Slice Templates (NSTs), and Network Service (NFV-NS) deployment flavor and instantation level given the specific set of values provided. These translation rules can be provided by the administrator during the VSB onboarding phase, or they can be managed to adapt the translation policies based on the monitoring information to adapt to the NFV infrastructure status (as part of the control-loops established in Sections 3.1.4 and 3.1.5). It is worth highlighting that, in 5Growth the vertical services include radio access network (RAN) slices as described in I1, which is detailed in Section 3.1.1. As also shown in Figure 2, in the final design, the 5Gr-VS supports the use CSMF and NSMF provided by other administrative domains to deploy Vertical (sub)services and Network Slices, based on the outcome of the translation process (we refer to Section 3.1.6 for further details).

Once the translation process determines the specific dimension of the network slices required, the CSMF functionality of the 5Gr-VS uses vertical service arbitration procedures, that allow determining the distribution of the infrastructure resources among the concurrent VSIs. In practice, arbitration

algorithms determine which NSIs can be shared among the different VSIs and if any vertical service lifecycle management action is required to fulfil the vertical SLA. The 5Gr-VS enhanced arbitration procedures enable the usage of AI/ML-based algorithms to compute the arbitration decisions (as described in Section 3.1.5).

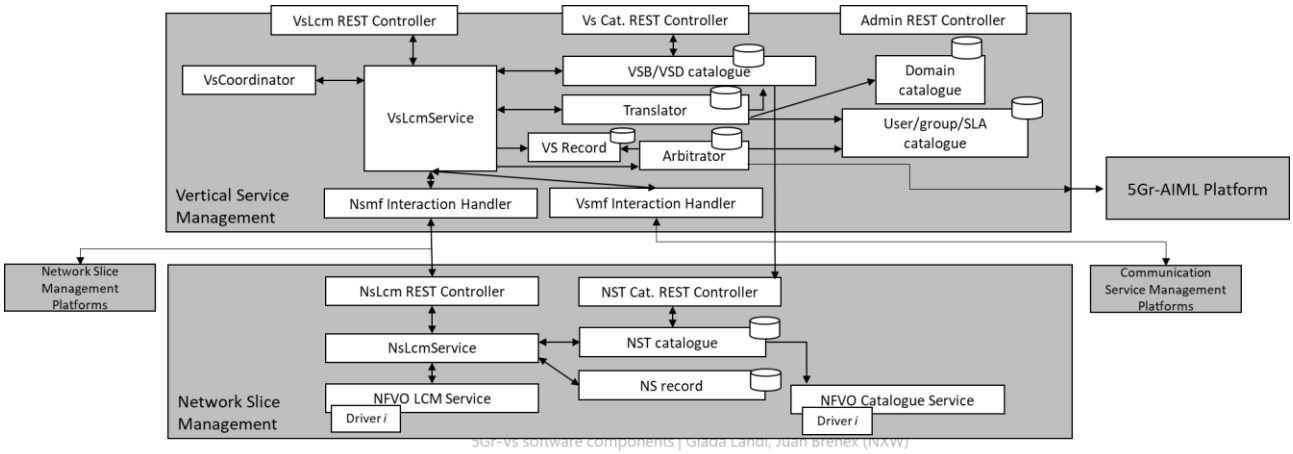


FIGURE 3: 5GR-VS FUNCTIONAL ARCHITECTURE

Figure 3 shows the functional architecture and blocks of the 5Gr-VS. At the northbound, the Vertical Slicer offers the following REST-based APIs (exposed also through a GUI):

- "VS Cat. REST controller" for VSB, VSD and translation rule management operations.
- "Admin REST controller" which supports tenant, SLA, and domain management requests.
- "VsLcm REST controller" which enables vertical service lifecycle management operations.

The full specification of the 5Gr-VS northbound interface in OpenAPI format is available in [97].

At the southbound interface, the 5Gr-VS supports interfaces with:

- 5Gr-SO, following the ETSI GS NFV-IFA 013 [5] and ETSI GS NFV-IFA 014 [6] for the interface definition and NFV-NS information models. This interface enables the management of NSDs, VNFDs and PNFDs associated with the network slices of the vertical services.
- Network Slice Management Platforms, by means of drivers that follow the 3GPP TS 28.531 [16] specification. These drivers are used by the "Nsmf Interaction Handler" to actuate on network slice requests, hiding the platform specific logic.
- Communication Service Management Platforms, using platform specific drivers at the "Vsmf Interaction Handler" block to request and manage vertical (sub)services.

The Vertical Service Management of the 5Gr-VS includes the functional blocks corresponding to the CSMF of the network slicing architecture. The *VsLcmService* is the core block of the platform since it is the responsible for interacting with the other blocks to process the vertical service lifecycle management request. The logic to map the vertical services to network slices and to the available NSIs is included in the "Translator" and "Arbitrator" blocks. The translator contains the logic to map between vertical services into network slices, establishing the network slice templates, NFV-NS and their specific deployment size based on the VSB and the VSD of the requested service. The arbitrator logic establishes which network slices are to be instantiated and which to be re-used from the already

available (and if scaling or modification operations are to be performed). As described previously, this block supports the retrieval and execution of arbitration AI/ML-based models available at the AI/ML platform. The outcome of the arbitration process is used by the "VsLcmService" to determine the requests to be issued to the Vsmf/Nsmf Interaction Handlers.

Similarly, the Network Slice Management contains the 5Gr-VS functional blocks corresponding to the NSMF and NSSMF of the slicing architecture. The core component in this case is the "NsLcmService" which interacts with a network slice template (NST) catalogue and with the NFVO Services ("NFVO LCM Service" and "NFVO Catalogue Service") to actuate upon network slice lifecycle management actions, using these latter services to query and manage NFV-NS.

2.2 Service Orchestrator

The 5Growth project implies an evolution of all the elements forming the 5G-TRANSFORMER stack. In particular, the 5Gr owth Service Orchestrator (5Gr-SO) maintains all the main features of the 5G-TRANSFORMER SO (5GT-SO), and introduces new ones to support the innovations introduced by the 5Growth project and to interact with the new blocks/elements of the 5Growth stack, like the AI/ML platform or the data pipeline management.

The 5Gr-SO, like its predecessor, the 5GT-SO, mainly oversees the coordination of the end-to-end orchestration of NFV-NSs and performs their lifecycle management. All of this is done across single or multiple administrative domains by interacting with the "local" 5Gr-RL (with northbound/southbound interface) and/or with other 5Gr-SOs of neighbor administrative domains (eastbound/westbound interface), respectively. In this last case, the interaction with peering 5Gr-SOs motivates the concept of federation, which maps the end-to-end deployment of network services in the most suitable execution environment based on the offered services and/or available resources in each administrative domain.

In addition, the 5Gr-SO adds new features (introduced through Releases 1 and 2 [3]) and mostly related to the innovations I1, I2, I4, I5, I6 and I10. This reflects an enhancement of the whole framework, algorithms, and architectural approaches regarding monitoring, forecasting, and smart orchestration capabilities.

For Innovation 1 (RAN segments in network slices), the support of the RAN segment management for network slices allows the 5Growth platform to ensure truly end-to-end QoS. In this sense, the northbound interface between the 5Gr-VS and the 5Gr-SO has been enhanced to support these slice parameters as well as the onboarding of the PNF descriptors. See Section 3.1.1.

For Innovation 2 (Vertical-service monitoring), the 5Growth Vertical-oriented monitoring system (5Gr-VoMS) has been developed as a Monitoring Platform inherited from the 5G-TRANSFORMER Monitoring Platform with enhanced extensions to increase its functionality. The work done on the 5Gr-SO side enhances the support to the 5Gr-VoMs by the installation of the Remote Virtual Machine (RVM) agents during the NFV-NS instantiation phase. For further details, see Section 3.1.2.

For Innovation 4 (Control-loops stability), the main features developed in the 5Gr-SO support a closed-loop operation to handle the AI/ML-based scaling of an NFV-NS. To accomplish this feature enhanced operations have been developed in several of the baseline blocks of the 5Gr-SO, like extending the SLA Manager and the Monitoring Manager blocks allowing the 5Gr-SO to fully support AI/ML-driven orchestration decisions, and consequently enabling Innovation 5 (AI/ML support). Moreover, the 5Gr-SO architecture also embeds an instance of Apache Spark and the AI/ML repository (included in the NFVI Resource Repository explained below), where the SLA Manager stores the requested AI/ML models and processing routines downloaded from the AI/ML Platform. See Section 3.1.4.

Other Innovations where the 5Gr-SO has been directly involved are Innovation 6 (Federation and inter-domain), to support the interaction with 5G-EVE Platform and the scaling of composite NFV-NS deployments involving multiple administrative domains, enabling the link between Innovation 6 and Innovation 4 (see Sections 3.1.6 and 3.1.4, respectively), and the Innovation 10 (Forecasting and inference), see Section 3.2.3.

2.2.1 Functional architecture

Figure 4 presents the functional architecture of 5Gr-SO building blocks and their interactions with the other components of the 5Growth stack. The inner structure of the 5Gr-SO is backwards compatible with the 5GT-SO and is designed to achieve the enhanced operation described before. It is also worth remarking that the 5Gr-SO architecture follows ETSI NFV guidelines.

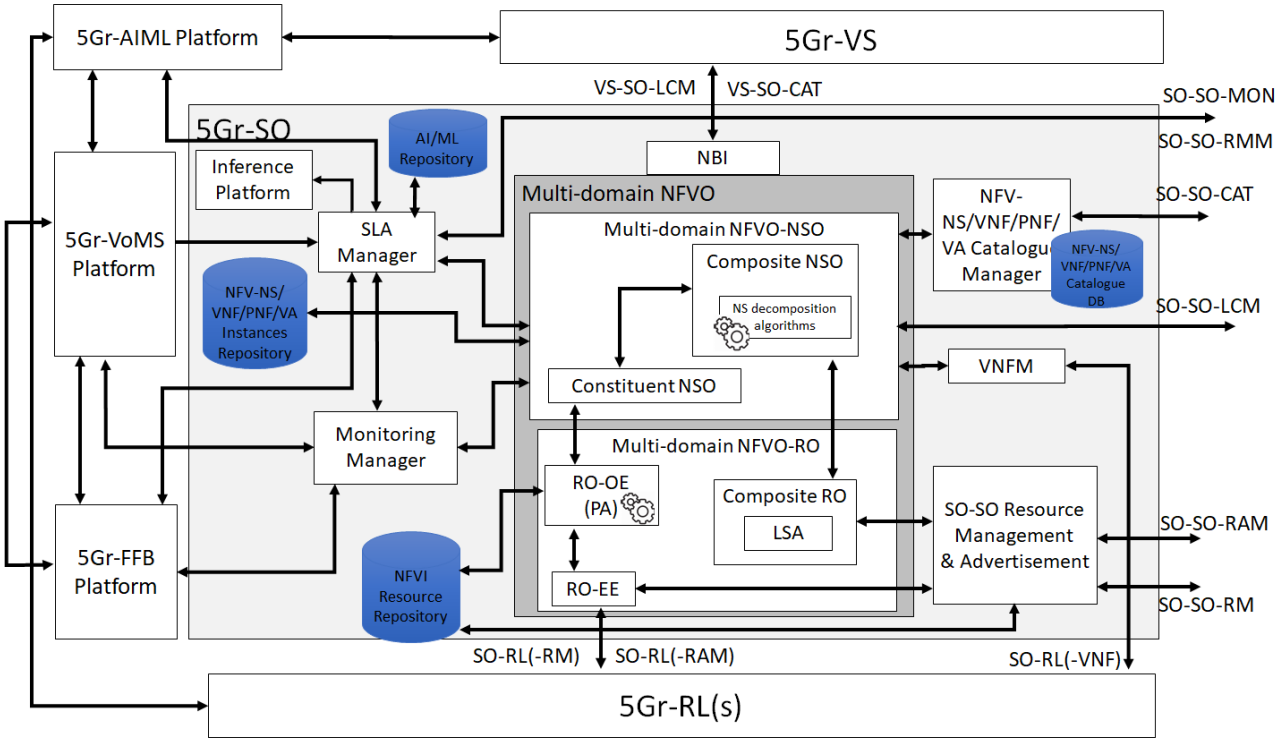


FIGURE 4: FUNCTIONAL ARCHITECTURE OF 5GR-SO

The main building blocks forming the 5Gr-SO are:

- **NBI:** This layer offers a northbound API towards the 5Gr-VS to support requests for service on-boarding, service instantiation, service modification (e.g., scaling), and service termination.
- **NFV Catalogue DB and Manager:** The Catalogue Manager offers access to the NFV Catalogue DB. This Catalogue DB is the repository for all the onboarded network service (NFV-NS), virtual network function (VNF), physical network function (PNF) and vertical application (VA) entities, represented according to ETSI IFA 011/014 descriptors [21][37]. As for the 5Gr-SO, those descriptors are used to process all the different phases of the lifecycle of the NFV-NS/VNF/PNF/VA. The Catalogue Manager also contains the aggregated information on the available NFV-NSs/VNFs/PNFs/VAs offered to the federated domains in case of the federation process. The aggregated information is stored by the Composite Network Service Orchestration (NSO).
- **Multi-Domain NFV Orchestrator (NFVO):** This block orchestrates virtual resources across local and inter-domain scenarios, performs the Resource Orchestration (NFVO-RO) functions and coordinates the deployment of the NFV-NSs along with their lifecycle operations. This block is sub-divided into two main sub-blocks, more in detail:
    - **Multi-Domain NFVO-NSO:** This sub-block coordinates all the operations about the deployment of the NS.
    - **Multi-Domain NFVO-RO:** This sub-block maps the NFV-NS into a set of virtual resources through the RO Orchestration Engine (**RO-OE**). The decision of where these virtual resources are placed is done by the RO-OE leveraging the placement algorithms (PAs). Such algorithms, based on heuristics, are used to attain the most efficient placement of the VNFs/PNFs/VAs using computing, storage, and networking resources fulfilling the NFV-NS requirements.
- **VNF Manager (VNFM):** This entity handles the lifecycle management of the VNFs deployed by the 5Gr-SO. According to the generic VNFM functionality defined by ETSI NFV, this is integrated into Management and Orchestration (MANO) platforms. The local NFVO block sends VNF lifecycle events, and it provides VNF reconfiguration according to the specified actions decided by the NFVO based on VNFDs (e.g., auto-scaling).
- **SO-SO Resource Management & Advertisement:** This block handles the communication with other domain's 5Gr-SO targeting resource Federation scenarios. The exchanged information is stored as federated resources into another local block called NFVI Resource Repository.

  With regards to the Service Federation, it only exchanges the related information about resources for inter-domain inter-nested path setup with the Composite RO.
- **NFVI Resource Repository:** This repository stores abstract resource information received from the underlying infrastructure. This can be either from the southbound interface (SBI) retrieving the information by the 5Gr-RL in case of local domain or from the eastbound/westbound Interface (EBI/WBI) retrieving the information by SO-SO Resource Management & Advertisement block in case of other federated domains..

- **NFV-NS/VNF/PNF/VA Instance Repository**: This repository stores the instances of VNFs and PNFs, NFV-NSs, and VAs that have previously been instantiated.
- **AI/ML Repository**: This repository has been added to store the downloaded AI/ML models from the 5Gr AI/ML Platform supporting the AI/ML-based scaling operations.
- **Monitoring Manager**: This block communicates with the Monitoring Platform (5Gr-VoMS). After retrieving the information from the elements of the underlying infrastructure, it activates (and takes care of their lifecycle) the "monitoring jobs" in the Monitoring Platform. The Monitoring Manager exchanges information regarding the "monitoring jobs" with the SLA Manager in order to enable the alerts and dashboard configuration, that allows the activation of notifications in the 5Gr-SO.
  In addition, this block oversees the installation of the RVM agent into VNFs in coordination with the 5Gr-VoMS. Specifically, during the instantiation phase of a Virtual Machine (VM), this block makes a request to the monitoring platform to "create RVM agent". The monitoring platform responds with a "cloud-init" file script for the specific VM. Finally, thanks to this cloud-init script, the RVM agent is created inside the VM and the 5Gr-SO can install exporters and remotely execute commands to the VM through the RVM agent during the instantiation, scaling and termination phase.
- **SLA Manager:** This block is in charge of preserving the SLAs between the 5Gr-VS and the 5Gr-SO concerning the deployed services. To this end, this block communicates with the 5Gr-VoMS. After a subscription phase, it registers specific monitoring parameter queries (retrieved by the monitoring jobs created by the 5Gr-SO Monitoring Manager) and creates a system of notifications related to specific threshold values of the defined monitoring parameters. Whenever a threshold is exceeded, the SLA Manager receives a notification and triggers consequent actions inside the 5Gr-SO.
  With the introduction of the 5Gr AI/ML Platform, the functionality of the SLA Manager has been further extended to handle all the operations of the AI/ML-based scaling operation if the appropriate information element is included in the NSD. For this purpose, the inference operation using the downloaded AI/ML model is carried out within the 5Gr-SO using Apache Spark. This approach follows the architectural guidelines of the O-RAN specification where the model training and inference jobs are performed in different building blocks.

2.3 Resource Layer

The 5Gr-RL follows the paradigm of NFV Infrastructure as a Service (NFVI-aaS). It is responsible for the orchestration of resources and the instantiation of VNFs and PNFs over the infrastructure under its control, as well as for managing the underlying physical mobile transport network, computing, and storage infrastructure. 5Gr-RL manages all the infrastructures as a Single Logic Point of Contact (SLPOC) [7] and is connected to different plugins that differentiate according to the infrastructure type under their control: the transport Wide Infrastructure Manager (WIM) plugins, the Virtual Infrastructure Manager (VIM) plugins, the radio Infrastructure manager (RADIO) plugins and ETSI-compliant Multi-Access Edge Computing (MEC) plugins. In addition to the SLPOC architecture

defined in the standard, 5Gr-RL provides a common abstraction view of the managed resources to the Service Orchestrator via the ETSI NFV IFA005 interface [8].

2.3.1 Functional architecture

The functional architecture of the 5Gr-RL is shown in Figure 5.

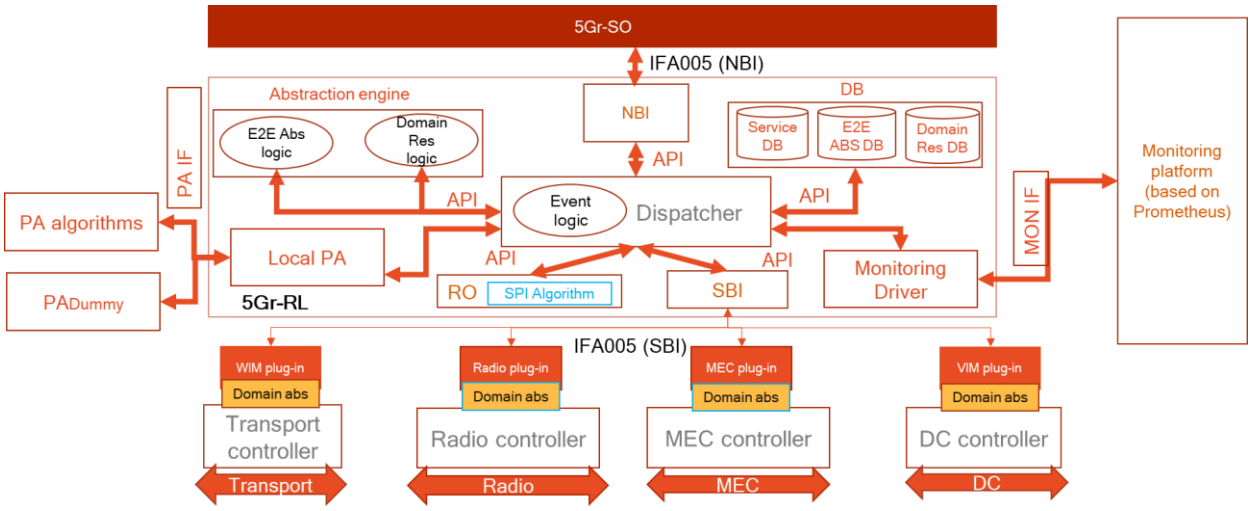


FIGURE 5: 5GR-RL ARCHITECTURE

The implementation of the new features defined in the 5Growth innovations requires extensions to the existing modules and the insertion of a new one. In blue, it is highlighted the new block added called the Slice Performance Isolation (SPI) Algorithm. This block is included as a submodule inside the Resource Orchestrator (RO) since it is integrated in the resource orchestration workflows of different physical infrastructure. Specifically, it is invoked by the RO during the resource allocation process to compute the QoS policy that needs to be applied in the physical infrastructure. This allows conducting the slice isolation of the resources as detailed in Section 3.2.1.2.2.

Other targeted innovations require the extensions of existing ones. The extended modules are:

- **Abstraction Engine**: this module implements all the algorithms and procedures related to the abstraction. It is extended to include the geo-location resource information and the management of PNF.
- **Database (DB)**: it is a front-end connecting to an external SQL server that stores all the information and details regarding the domain resources, the abstraction view, and the allocated services. The DB and SQL servers are extended to include information regarding the PNF management and QoS Policy used for slice performance isolation.
- **Dispatcher**: it is an event bus that manages the inter-process communication between all the 5Gr-RL components. This module is extended with new workflows and communications to handle PNF, RAN slice and Slice performance isolation features.
- **Resource Orchestrator (RO)**: it orchestrates the resource management between VIM, Radio, WIM, MEC domains. This module is extended to handle PNF resources and to handle the slice performance isolation features.

- **Northbound Interface (NBI)**: it handles the ETSI NFV IFA005 [8] communication with 5Gr-SO. Additional APIs are added to handle PNF.
- **Southbound Interface (SBI)**: it handles the ETSI NFV IFA005 [8] communication with the plugins. Additional APIs are added to handle PNF, RAN slice and Slice Performance Isolation.
- **RL Plugin**: Additional plugins are added/ extended to support WP3 pilots and innovations. Such additional plugins will be described in D2.4 [9].

2.4 AI/ML Platform

The AI/ML Platform (5Gr-AIMLP) realizes the concept of AI/ML as a Service (AIMLaaS), thus addressing the need for AI/ML models for fully automated service management, network orchestration, and resource control within the 5Growth architecture. This building block is related to the innovation 5 (see Section 3.1.5). Specifically, the 5Gr-AIMLP is a centralized and optimized environment for the efficient training, storage, and serving of AI/ML models that may be needed for any decision-making process at any layer of the 5Growth stack (e.g., for slice arbitration at the 5Gr-VS, for automated NFV-NS scaling at the 5Gr-SO, for automated path restoration at the 5Gr-RL, or for forecasting at the 5Gr-FFB). The architecture and the fundamental workflow of the platform are depicted in Figure 6, along with the entities between which the main interactions take place.

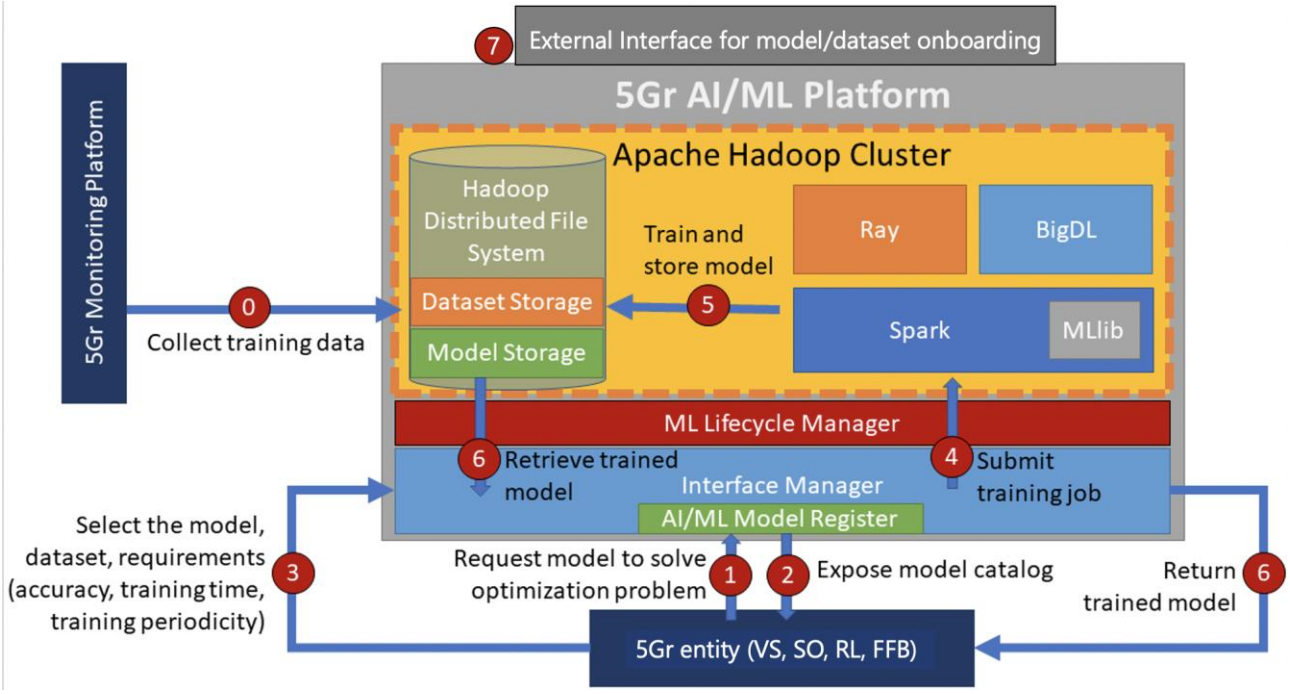


FIGURE 6: STRUCTURE AND WORKFLOW OF THE AI/ML PLATFORM

Such entities are the 5Gr-VoMS (Section 2.5) and a generic entity of the 5Growth architecture requiring a trained model (e.g., the 5Gr-VS or 5Gr-SO), hereinafter simply referred to as 5Gr-entity. The 5Gr-VoMS provides raw monitoring data that is used for taking operational decisions once the model is running in the 5Gr-entity and are also collected at the 5Gr-AIMLP to build training datasets, which can be subsequently used for training purposes. The 5Gr-entity also interacts with the 5Gr-

AIMLP, in particular with the Interface Manager, to request and receive AI/ML models trained on the latest dataset available.

The 5Gr-AIMLP includes the following main components:

- **Model Registry**, which records the models uploaded to the platform, their metadata, and pointers to the stored models and associated files.
- **Lifecycle Manager**, which is in charge of the models' lifecycle. Upon uploading a new model, it adds the corresponding entry to the Model Registry and, if it is an untrained model, it triggers the training process using the appropriate AI/ML framework. After a model is trained, the Lifecycle Manager monitors its status: it can trigger a new training job either periodically, or whenever new data are available from the 5Gr-VoMS.
- **Interface Manager**, which processes the requests for AI/ML models coming from the architectural stack and forwards them to the proper block inside the computing cluster
- **Computing cluster**, which is based on Apache Hadoop [10] and leverages Yet-Another-Resource-Negotiator (YARN) for the computing resources management, and the Hadoop Distributed File System (HDFS) for the storage of datasets and models. The YARN cluster nodes have access to different AI/ML frameworks, according to the requested model type. Spark [11] is used to train classic supervised and unsupervised models, BigDL [12] is used for Deep Neural Networks, and Ray [13] can be used for Reinforcement Learning models.

The envisioned workflow, as also depicted in Figure 6, is set forth below:

0. Monitoring data from the 5Gr-VoMS is collected, reformatted, and consolidated to build or update a training dataset. Training datasets are then saved in the HDFS Dataset Storage.
1. 5Gr-entities trigger optimization processes (e.g., placement, SLA management) to support service lifecycle management operations (e.g., deployment, scaling) of a vertical service (and its underlying network service). Alternatively, 5Gr-entities may also want to optimize resource usage of infrastructures under their control or may decide to react on some detected anomaly. When deploying such optimization processes, the 5Gr-entity will request to the 5Gr-AIMLP the available model catalogue that suits each of the problems at hand.
2. At this stage, the 5Gr-AIMLP offers a set of available models, which include already trained models as well as models that can be trained on-demand, as indexed by the Model Register.
3. The 5Gr-entity selects the model, and may specify some requirements, like accuracy and training time. The training periodicity or an accuracy threshold can be optionally set, e.g., to let the Lifecycle Manager automatically keep the model fit.
4. In case the selected model requires preliminary training, either because it has never been trained or its validity has expired, a training job is submitted to YARN. If the requested model is valid, it is directly fetched from the Model Storage.
5. Using the proper dataset from the Dataset Storage, the training job is performed using either Spark MLlib, BigDL or Ray, depending on the model type. Once the training is complete, the trained model is saved in the Hadoop Distributed File System (HDFS) Model Storage. The ML

Lifecycle Manager tracks the new trained model state and updates the Model Register accordingly.

6. The trained model is finally retrieved from the HDFS and returned to the requesting 5Gr-entity, which is responsible for its online execution.
7. In case the selected model requires preliminary training, either because it has never been trained or its validity has expired, a training job is submitted to YARN. If the requested model is valid, it is directly fetched from the Model Storage.

Using the proper dataset from the Dataset Storage, the training job is performed using either Spark MLlib, BigDL or Ray, depending on the model type. Once the training is complete, the trained model is saved in the HDFS Model Storage. The ML Lifecycle Manager tracks the new trained model state and updates the Model Register accordingly. The trained model is finally retrieved from the HDFS and returned to the requesting 5Gr-entity, which is responsible for its online execution.

Finally, we underline that, through the web interface (marked in the figure as 7), an authorized external user can also onboard onto the AI/ML platform off-line trained ML models, as well as ML models and the corresponding datasets, to be trained within the platform itself.

2.5 Vertical-oriented Monitoring System

The Vertical-oriented Monitoring System (5Gr-VoMS) is a part of the 5Growth stack. It is a conglomerate of software components and logical processes, combined under a single architecture, that serves the purpose to observe and gather logs and metrics from the vertical application workloads. . This building block is related to the innovation 2 and 3 (see Section 3.1.2 and 3.1.3). In a nutshell, the 5Gr-VoMS is an evolution of the 5G-TRANSFORMER Monitoring Platform (5GT-MTP). Comparing to its ancestor, it is much more advanced since it was enhanced with more vertical-oriented features like configurable probes (a small VNF-side monitoring agent) for gathering monitoring data from the vertical VNFs or logs monitoring. A more detailed description of the 5GT-MP architecture can be found in 5G-TRANSFORMER D4.3 [14]. Also, 5Gr-VoMS is formerly known as the 5Growth Monitoring Platform[2].

The architecture of the 5Gr-VoMS is shown in Figure 7. Please note that this architecture diagram shows the 5Gr-SO as the main and the only service that communicates with the 5Gr-VoMS. But this is not exactly so. It is shown in such a way only to illustrate the 5Gr-VoMS configuration process. From the other perspective, 5Gr-VoMS can be accessed by any service through its API to retrieve monitoring data.

[2] Notation: To avoid misunderstanding and compatibility with previous documentation, the old naming (5Gr-MP) corresponds to the new 5Gr-VoMS.

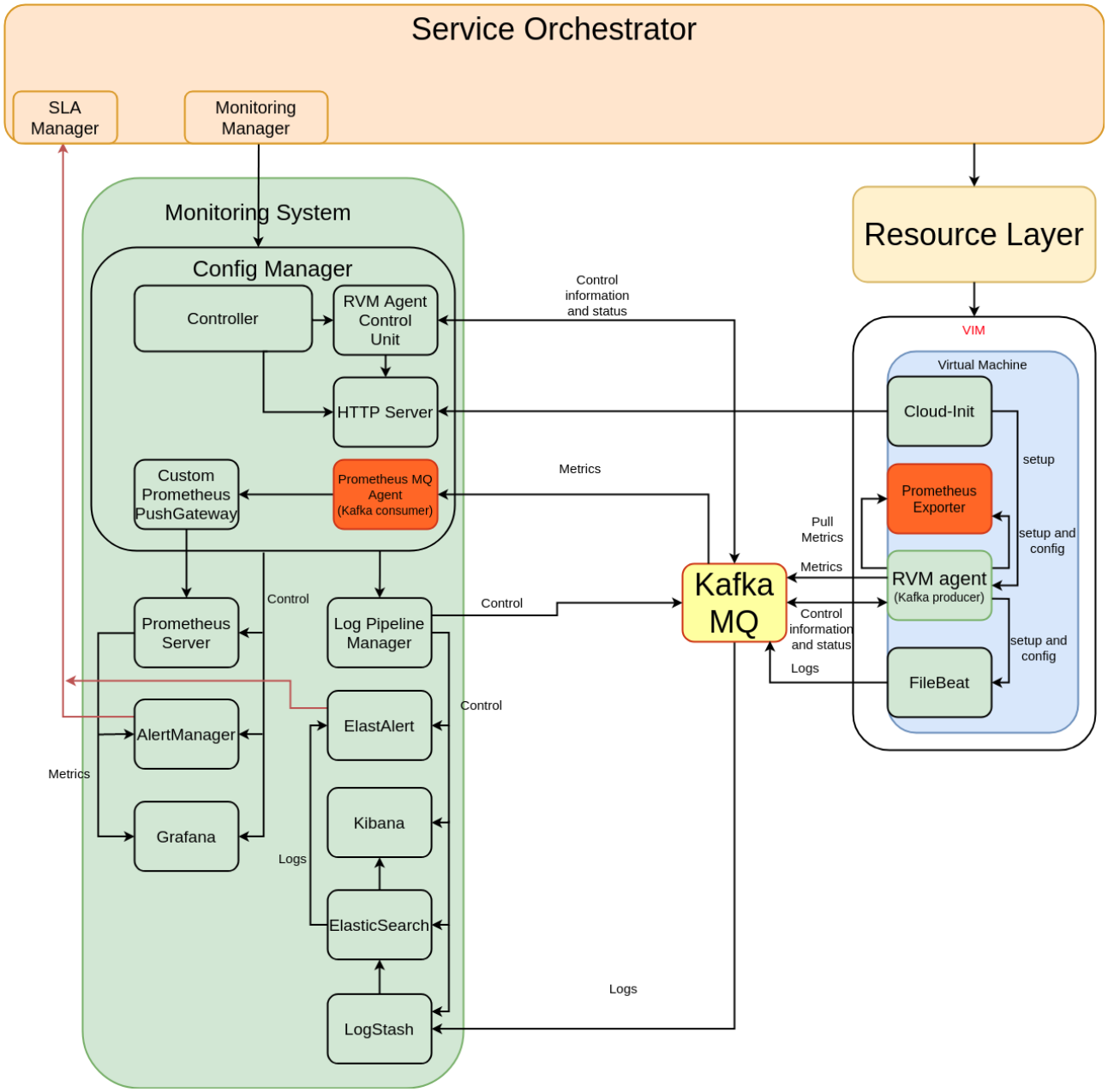


FIGURE 7: VERTICAL-ORIENTED MONITORING SYSTEM ARCHITECTURE

5Gr-VoMS includes the following elements:

- **Config Manager**: it is the main block of the Monitoring System that receives control requests from 5Gr-VS, 5Gr-SO, 5Gr-RL and managing all elements in the 5Gr-VoMS.
- **Prometheus Message Queue Agent (MQAgent)** is a part of the config manager that gathers metrics from Kafka Topics [100] and sends them to Custom Prometheus Push Gateway.
- **Custom Prometheus Push Gateway** is responsible for transferring the received metrics into the Prometheus server. Initially, the Open-Source Prometheus Push Gateway [18] was used as part of 5Gr-VoMS but it has disadvantages. Native Open-Source Prometheus Push Gateway component has the same scrape interval for all metrics, scrapping interval cannot

be changed individually for each metric, and it is unable to detect whether the Prometheus exporter is unavailable. To solve this limitation, Custom Prometheus Push Gateway has been developed.

- The **RVM agent control unit** is responsible for the installation of RVM agents at remote VMs and interaction with them through Kafka Topics.
- **HTTP server** is responsible for the distribution of the RVM agent installation init-scripts, RVM agent itself, and monitoring probes.
- **Prometheus Server** is an open-source platform that collects metrics from monitored targets. Prometheus data is stored in the form of Time Series Database (TSDB). Each metric has a name that is used for referencing and querying it. Prometheus stores data locally on disk, which helps for fast data storage and fast querying.
- **Grafana** provides graphs and visualizations of metrics. Grafana is multi-platform open-source analytics and interactive visualization web application. It provides charts, graphs, and alerts for the web when connected to supported data sources.
- **AlertManager** processes Prometheus alerts. AlertManager takes care of deduplicating, grouping, and routing alerts to the correct receiver.
- The **RVM agent** is a software component that was developed for the 5Gr-VoMS. It uses Kafka as a Message Queue for interaction with the 5Gr-VoMS. The RVM agent can execute bash and python scripts at a remote virtual machine. This gives the possibility to change VM configuration and install extra software. The VM agent is used for monitoring probes (exporters) dynamic installation and configuration. Also, the RVM agent changes pull Prometheus architecture to push architecture. The RVM agent can receive instruction about the creation of a "Prometheus collector" from 5Gr-VoMS. The software entity "Prometheus collector" can be created inside the RVM agent. Prometheus collector receives the configuration from the 5Gr-VoMS, it performs requests to installed Prometheus Exporter for metrics and publishes these metrics to Kafka Topic. The advantage of this RVM agent architecture, in comparison with existing configuration management systems like Puppet, Salt, or Ansible, is that it doesn't require a dedicated management network for interacting with the 5Gr-VoMS, as the RVM agent has control and data plane communication, performed through the Kafka broker.
- **Log Pipeline Manager** in charge of managing the configuration lifecycle of the components related to the log monitoring process.
- **Logstash** is the data collection pipeline tool. It collects data inputs and feeds them into Elasticsearch. It aggregates all types of data from different sources and makes it available for further use.
- **Elasticsearch** allows storing, searching, and analyzing big volumes of data. It is mostly used in these applications as the underlying engine for implementing search task functionality. It has been adopted in search engine platforms for modern web and mobile applications. Apart from a quick search, the tool also offers complex analytics and many advanced features.
- **Kibana** is used for visualizing Elasticsearch documents and helps developers to have a quick insight into it. Kibana dashboard offers various interactive diagrams, geospatial data, and

graphs to visualize complex queries. It can be used for search, view, and interact with data stored in Elasticsearch directories. Kibana helps to perform advanced data analysis and visualize the data in a variety of tables, charts, and maps.

- **ElastAlert** is a simple framework for alerting on anomalies, spikes, or other patterns of interest from data in Elasticsearch.
- **Kafka MQ** is a bus for metrics, logs, and control information for RVM agents.

The 5Gr-VoMS has an interaction also with 5Gr-FBB. In general, the forecasting block can be seen as a standard Prometheus exporter considering the forecasting values. In fact, dedicated Prometheus jobs are configured, in order to periodically retrieve forecasting data, to be locally stored.

A more detailed description of the 5Gr-VoMS can be found in 5Growth D2.4 [9].

2.6 Forecasting Functional Block

The Forecasting Functional Block is a module of the 5Growth platform whose services can be exploited by several modules of the platform. The purpose of the forecasting functional block is to generate forecast values of parameters monitored by the 5Gr-VoMS platform. The forecast values can be utilized by decision modules of the platform to take a proactive approach in reacting to changing status.

A sample interaction of the forecasting functional block architecture with the other 5Growth platform functional elements is depicted in Figure 8.

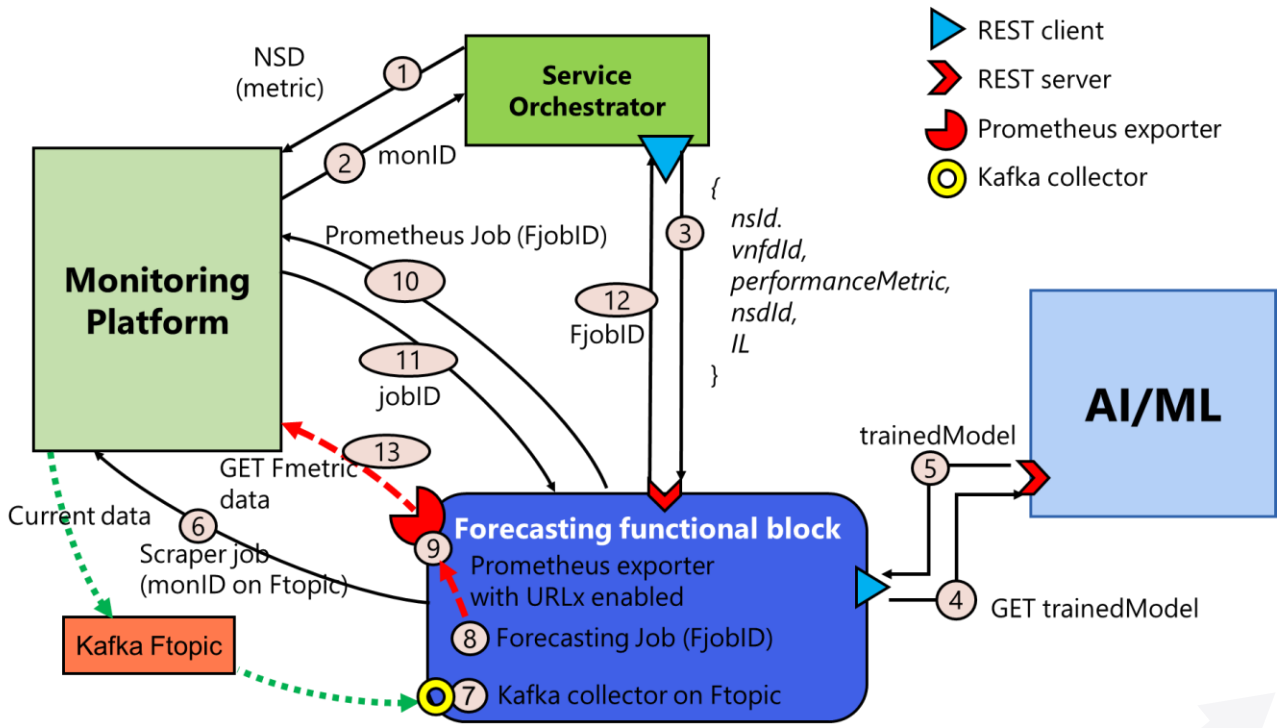


FIGURE 8: EXAMPLE OF INTERACTION OF THE FORECASTING FUNCTIONAL BLOCK WITH OTHER 5GROWTH PLATFORM FUNCTIONAL ELEMENTS.

In this case the forecasting functional block provides the service orchestrator forecast of the monitored data to be used to take, for example, pro-active service scaling decisions. This approach has the potential advantage of allocating the required resources just in time to minimize resource utilisation and SLA violations. The interaction between the 5Gr-FFB and the other elements is based on the following APIs:

- The API among 5Gr-SO and 5Gr-FFB enables the exchange of information among the two functional blocks about which parameter(s) must be forecast and how they are identified; moreover it allows to activate and deactivate parameter forecasting.
- The API among 5Gr-VoMS and 5Gr-FFB allows from one side the 5Gr-FFB to retrieve the current values of the parameters to be forecast (via Kafka consumer connected to a dedicated Kafka topic) and then to provide the forecast data to be made available through the 5Gr-VoMS to the other decisional elements (via dedicated Prometheus exporter plugin).
- The API among the 5Gr-AIMLP and the 5Gr-FFB allows the 5Gr-FFB to request the 5Gr-AIMLP trained model to be used for parameter forecasting; the interaction can be as simple as the retrieve of a model already trained offline with already available data or as complex as the request of a specific AI/ML model, the provisioning of data for model training and the retrieving of the trained model to perform forecasting.

A sample workflow of the considered interaction is the following. The NSD received by the 5Gr-SO specifies which parameter shall be forecasted. The 5Gr-SO, activates the monitoring of the specified metric in the 5Gr- VoMS(1) and it receives a monitoring ID (monID) back (2). The 5Gr-SO activates the Forecasting process in the Forecasting functional block passing some parameters, such as nsId, vnfdId, performanceMetric, nsdId and instantiation level (3). Then, 5Gr/FFB gets a trained model from the 5Gr-AIMLP (in general it can ask for training a model if needed) (4), (5) and it stores it locally. The forecasting functional block activates a new Kafka topic over the Kafka cluster, dedicated to the transmission of the values related to current data. Then, it activates a scraper job (6), filtering the data according to input parameters (i.e., nsId. vnfdId, performanceMetric), and enables a Kafka consumer in order to receive the monitoring metric data from the forecasting platform (7). The forecasting job (FjobID) is then started running the stored model and using as input parameters the metric values received via Kafka consumer (8). A Prometheus job is configured at the 5Gr- VoMS in order to periodically get the forecasting data related to the enabled forecasting job (identified by FjobID) using the Prometheus exporter interface, acting as a standard probe (9). Finally, the forecasting platform notifies the 5Gr-SO of the enabled forecasting job ID (FjobID) (12). The 5Gr monitoring platform will then retrieve forecast metrics through Prometheus (13). Such metric can be accessed by the 5Gr-SO to include them in the decisions to be taken.

3 5Growth Innovations

The above architecture has embedded a set of novel innovations and features aimed at covering the gaps motivated by the project pilot use cases [1]. We finally encoded our gap analysis into Table 2, which maps the feature requirements or gaps into a set of 12 innovations:

- I1: Support of Verticals. RAN segments in network slices
- I2: Support of Verticals. Vertical-service monitoring
- I3: Monitoring orchestration
- I4: Control and Management. Control-loops stability
- I5: Control and Management. AI/ML Support
- I6: End-to-End orchestration. Federation and Inter-domain
- I7: End-to-End orchestration. Next Generation RAN
- I8: Smart orchestration and resource control
- I9: Anomaly detection algorithms
- I10: Forecasting and inference
- I11: Security and auditability
- I12: 5Growth CI/CD

TABLE 2: MAPPING BETWEEN PLATFORM REQUIREMENTS (GAPS) AND SELECTED INNOVATIONS

Feature (gap)	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10	I11	I12
Enhanced VS network slice sharing				X	X							
VS arbitration at runtime		X	X	X	X			X				
VS layer federation						X						
VS dynamic service composition				X		X		X				
SO automatic network service management		X	X	X	X			X	X	X		X
SO self-adaptation actions		X		X	X	X		X	X	X		
SO dynamic monitoring orchestration		X	X	X					X			
SO geo-location dependent federation				X		X						
RL PNF integration	X						X					
RL geo-specific resources	X											
RAN support	X						X		X			
Integral security											X	

Continuous development and integration												X

In the following, we present in detail the design of each of the aforementioned innovations, and explain how they contribute to fill the designated gap.

3.1 Architecture Innovations

This section presents those innovations related with the application of novel concepts to the architecture of 5Growth.

3.1.1 Innovation 1 (I1): Support of Verticals. RAN segments in network slices

As described in D2.1 [1], a typical vertical service instance (VSI) maps to network slices spanning from the radio access network (RAN) to the core network passing through the transport network, as shown in Figure 9 (where VSI are referred to as Communication Service Instances). This means that a VSI traverses the operator's network connecting the end user equipment with the vertical service application service logic placed arbitrarily at any place of the network. The resources contained in these segments (i.e., RAN, transport and core) have different characteristics and therefore the way to model and request the allocation of the resources is different.

On the one side, the core segment resources of the network slice are determined by the NFV-NSs of the vertical service and the deployment dimensions. This resource allocation is typically handled by the traditional Network Function Virtualization (NFV) management and orchestration (MANO) as part of the NFV-NS lifecycle management.

On the other side, the radio access resources mostly depend on the mobile traffic category and characteristics of the vertical service. It should be noted that different service slice types (*enhanced Mobile BroadBand (eMBB)*, *Ultra-Reliable Low-Latency Communication (URLLC)* and massive *Machine Type Communication (mMTC))* can be associated to specific subset of parameters at the slice profile level, which impact the mobile traffic and therefore the access network configuration in different ways. Examples of common parameters are area traffic capacity, coverage area, overall user density, service availability, service reliability, etc., while others like end-to-end latency are specific to a given service slice type (e.g., URLLC network slice in this case). The allocation and configuration of the RAN resources is handled by specific RAN controllers, which interact with mobile infrastructure devices (i.e., Base Band Units (BBUs), Remote Radio Heads (RRHs), etc).

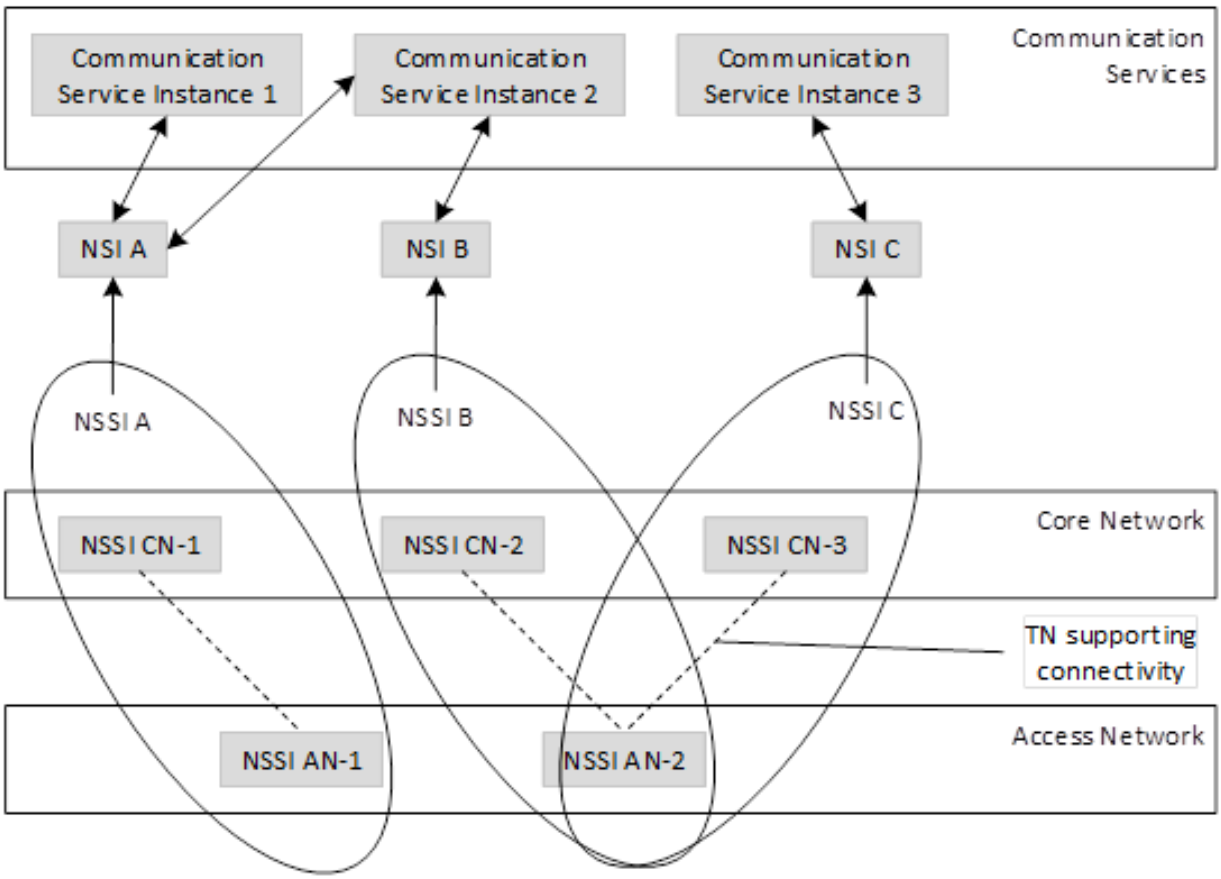


FIGURE 9: COMMUNICATION SERVICES PROVIDED BY NETWORK SLICE INSTANCES WITH CORE AND ACCESS NETWORK SLICE SUBNETS [22]

In 5Growth, the vertical services have been enhanced to include RAN slices as part of the end-to-end network slice of the service, therefore enabling the support for vertical services with specific RAN characteristics. The 5Gr-VS, 5Gr-SO and 5Gr-RL information models and interfaces have been updated in order to support the definition of the vertical service RAN characteristics and the allocation and configuration as part of the vertical service deployment. The high-level workflow is illustrated in Figure 10.

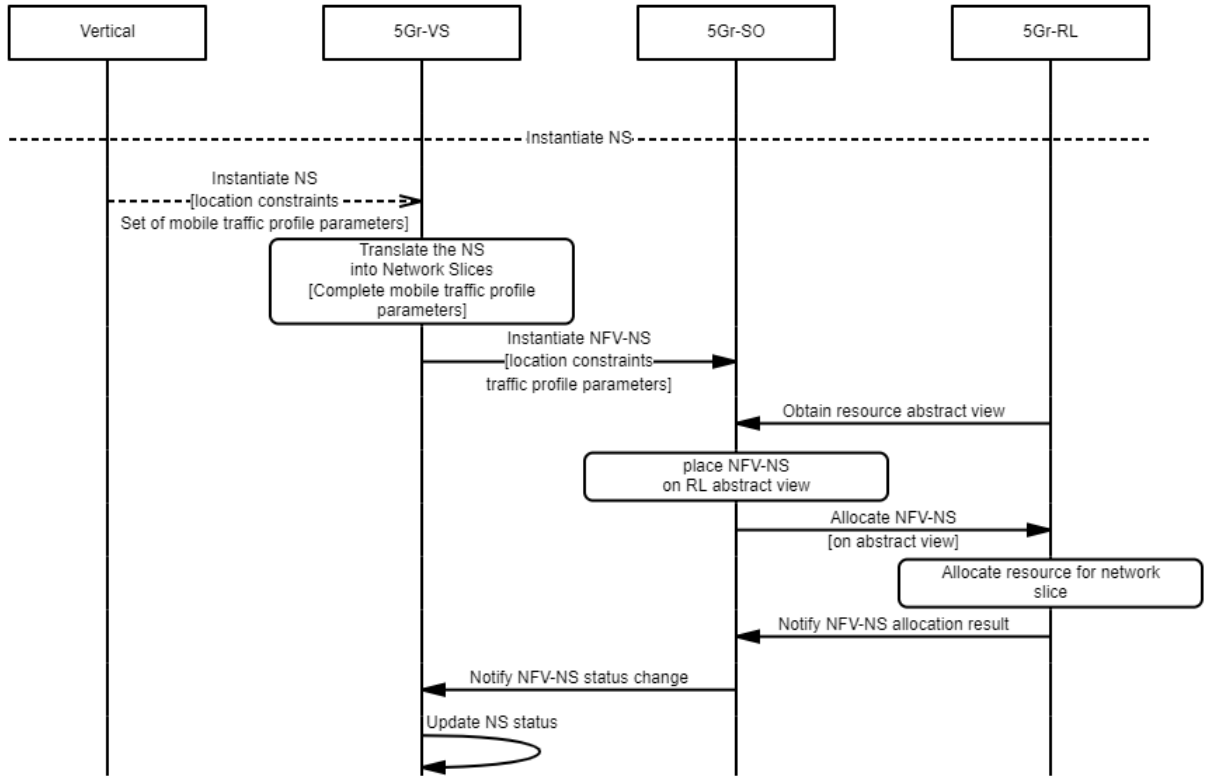


FIGURE 10: VERTICAL SERVICE INSTANTIATION WORKFLOW WITH RAN SLICE

There is an ongoing standardization effort targeting to include the RAN segment as part of the network slices. In 3GPP, the slice management architecture is defined and enhancements to use the slice feature attributes defined in GSMA NG.116 [27] are being undertaken. In the 3GPP slicing architecture, TS28.530 [22] specifies the management system and requirements for network slices, TS28.531 [16] defines interfaces, TS28.541 [24] specification defines the network resource model for network slicing and the SLA requirements related to end-to-end slicing, TS23.501 [25] and TS23.502 [26] define the relevant architectures and procedures of core network element support slice, and TS38.300 defines the RAN side network slicing principles, slice selection, processing, mobility and other signalling procedures. 3GPP is planning to use the slice feature attributes defined in GSMA NG.116 [27] and is working on a new report about it.

3.1.1.1 5Gr-VS enhancements

At the 5Gr-VS level, the support of RAN slices required updating the information models of the Vertical Service Blueprints (VSBs) and Vertical Service Descriptors (VSDs) in order to let the 5Gr-VS determine mobile traffic category of the vertical service and to allow the vertical to customize the specific value required for each traffic characteristic, as follows:

- The VSB information model was updated to include a *sliceServiceType* enumerator accepting *eMBB, mMTC or URLLC*. A *serviceCategory* was added to establish a service category which allows retrieving the default values for the mobile traffic characteristics (as defined in 3GPP TS 22.261).

- The VSD information model was updated with a *sliceServiceParameter* that allows overriding the default values determined by the *serviceCategory* of the VSB, allowing the vertical to customizing the vertical service to its particular requirements.

During the vertical service deployment, the 5Gr-VS computes the final list of mobile traffic characteristics based on the VSB, VSD information, and the translation rules. This information is included in the NFV-NS Instantiation request which is sent to the 5Gr-SO as part of the service access point (SAP) data (*sapData)*, associated to the SAP accessible from the RAN (using an updated VS-SO interface).

This innovation also required the support Physical Network Function Descriptors (PNFDs) onboarding, in order to include the mobile infrastructure equipment as part of the NFV-NS definition. Within this scope, the VSB onboarding request of the 5Gr-VS was updated to support the inclusion of the PNFDs to be used. The 5Gr-VS triggers then the onboarding of the PNFDs on the 5Gr-SO as specified in the next section.

3.1.1.2 5Gr-SO enhancements

The support of RAN segments in network slices implies several changes at the 5Gr-SO level to include the management of PNFs enabling the match of the requests of the 5Gr-VS at the NBI with the abstracted resource representation (e.g., list of supported PNF and RAN capabilities of available NFVI-PoP) provided by the 5Gr-RL at the SBI.

The NBI of the 5Gr-SO has been extended to support RAN-slice related parameters. This support is included inside the sapData information element of the ETSI NFV IFA013 [29] instantiation request. This information will be included for SAP elements corresponding to the service RAN endpoint, as explained previously. Additionally, the NBI offers support to satisfy PNFD onboarding requests from the 5Gr-VS. Table 3 presents the available endpoints supporting the PNFD onboarding functionality.

TABLE 3: EXTENDED 5GR-SO NBI ENDPOINTS FOR VNFD MANAGEMENT

Name	Method	Endpoint	Description
Get PNFD	GET	/ns/pnfd/{pnfdId}/{version}	Returns information of the Physical Network Function (PNF) referenced by *pnfdId* and *version* parameters
Post PNFD	POST	/ns/pnfdManagement/pnfd	Triggers the PNFD onboarding operation at the PNFD catalogue of the 5Gr-SO and returns information of the onboarded physical network function
Delete PNFD	DELETE	/ns/pnfd/{pnfdId}/{version}	Deletes the onboarded physical network function referenced by *pnfdId* and *version* parameters

Once PNFD management is enabled, the following considerations must be handled during the NFV-NS instantiation workflow to coordinate with the 5Gr-RL:

- The 5Gr-SO should parse the corresponding NSD and the instantiation request information to provide the Placement Algorithm (PA) with the needed information to select the suitable NFVI-PoPs hosting the required RAN-PNFs and the corresponding NFVI-PoPs to instantiate the VNFs present in the NSD. In this direction, the PA API needs to be able to distinguish the different nature, characteristics, and requirements of NFs in the NSD.
- VNFs will be instantiated first. After that, the 5Gr-SO will trigger the Physical Device Unit (PDU) allocation API of the 5Gr-RL (detailed next) to physically allocate the resources in the requested PNF. It is worth noting that the 5Gr-SO request has to include the slice related parameters to trigger the management of corresponding Management Functions (MF) (the basic element to configure Radio Equipment, see below) and the characteristics of the new virtual interface (IP address, VLAN information) that will be created at the PNF to enable the communication with required VNFs.
- Beside communications between VNFs, the 5Gr-SO has to handle the communication between required VNF-PNF pairs.

3.1.1.3 5Gr-RL enhancements

Several changes are done inside the 5Gr-RL layer to support the innovation "RAN segments in network slices". The main issue is that radio resources could not be virtual but formed by physical devices that have some firmware already up and running. Following ETSI MANO specifications, such devices require the handling of a new type of network function called Physical Network Function (PNF). In addition to that, RAN network slicing requires the application of radio specific commands to radio devices. PNF support leads to introduce extensions in 5Gr-RL in the abstraction of the radio physical resources and the management of resources for radio VNF/PNF. The application of radio specific commands for network slice configuration requires the handling of a new type of information through the radio Management Function (MF). These MFs are the basic element to configure Radio Equipment. It is defined in TS 28.622 [28] and represents a set of radio parameters (that could be standard or vendor specific) that is needed to apply in order to make the radio network function (both virtual and physical) work.

Regarding the abstraction of radio resources, the "Abstraction Engine" abstracts the radio domain resources using the "Radio Capable NFVI-PoP" defined in 5G-TRANSFORMER project [23] and adding additional PNF parameters that help the placement algorithm in 5Gr-SO to properly allocate the PNF inside the radio domain. Table 4 reports the list of PNF parameters used in the abstraction.
The management of VNFs is a feature already implemented in 5Gr-RL and used to handle Radio VNFs. This feature allows the module allocation/termination of the VMs running the applications and the relative connections between them. PNF management is a new feature that requires to extend the 5Gr-RL to handle the instantiation and termination of the PNF instance (referred to as PDU). Table 5 defines the new APIs use by 5Gr-RL in northbound interface (NBI) module to trigger the PDU instantiation and termination workflow, while

Table 6 defines the new APIs used by the 5Gr-RL in the southbound interface (SBI) module for the same operation.

TABLE 4: PNF PARAMETERS IN RADIO ABSTRACTED RESOURCES

Name	Description
PNFid	Identifier of PNF (is the same value used in the PNF Descriptors)
pdumaxnum	Maximum number of PNF instances supported by the radio Capable PoP
shared	Flag indicating if the PNF can be shared among multiple slices
type	Type of PNF (for example if it is a PNF inside the RAN access or Core)
physical location info	The geographical location of the PNF
Interface physical rate	Bitrate of the PNF interfaces (in Mbit/s)

Such parameters are exposed on the abstract view to 5Gr-SO to help the PNF placement on radio NFVI-PoP.

TABLE 5: NEW 5GR-RL NBI API FOR PNF MANAGEMENT

Name	Method	Endpoint	Description
NBI PDU Instantiation	POST	/physical-resources	Request allocation of PDU on abstract radio capable POP. It returns a reference ID to PDU resource
NBI PDU Termination	DELETE	/physical-resources	Request the termination of PDU previously allocated on a radio capable POP. The reference ID is used to pointing the PDU resources to be deallocated

TABLE 6: NEW 5GR-RL SBI API FOR PNF MANAGEMENT

Name	Method	Endpoint	Description
SBI PDU Instantiation	POST	/pnfs	Request allocation of PDU on radio domain.
SBI PDU Termination	DELETE	/pnfs	Request termination of PDU previously allocated.

In the PDU instantiation process, when the PDU instantiation request is received via NBI, the Abstraction Engine selects the radio domain and the geographic location of the resources to use for PDU allocation. This selection is passed to Resource Orchestrator that contacts the Radio plugin controlling the radio domain to provide and configure the required resource for PDU. The SBI PDU instantiation API is used by 5Gr-RL to send command to radio plugin. When the plugin gives a positive reply, 5Gr-RL provides a PDU reference id (PduRefId) to 5Gr-SO to refer such resource.

In the PDU termination process, when the PDU termination request is received via NBI, the PduRefId is passed in the request and used by Resource Orchestrator to retrieve the radio resources to deallocate in the radio domain. The SBI PDU Termination API is used by 5Gr-RL to send the command to radio plugin.

The last change in 5Gr-RL is the handling of radio Management Function (MF). 5Gr-RL implementation of such feature is based on the standard IFA024 [30] that specifies the relation between VNF/PNF and MF. Such relation is an association between PNF/VNF identifiers (the same reported in PNFD/VNFD) and the MF identifiers. To note that the association is 1:N, i.e., multiple MFs can be associated to a single PNF/VNF.

The implemented workflow is reported in Figure 11.

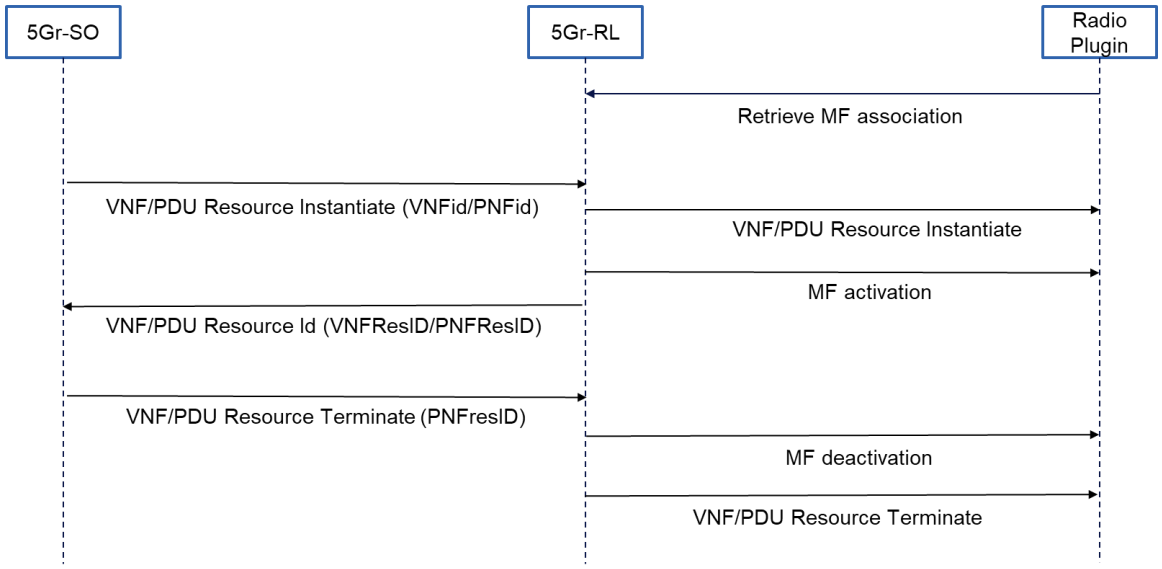


FIGURE 11: MF MANAGEMENT WORKFLOW IN 5GR-RL

As first step (that is done at 5Gr-RL bootstrap), 5Gr-RL requires all MFs that need to be applied when instantiated a radio VNF/PNF inside the radio domain. The association is retrieved using a new SBI API that is reported in Table 7.

When a VNF or PDU instantiation is requested by 5Gr-SO, the 5Gr-RL instantiates the VNF/PDU as described above and then asks to radio plugin the activation of MFs associated with VNF/PNF. The MF instantiation request is done using a new SBI API reported in Table 7.

In a similar way, when a VNF/PDU termination request is received via NBI, the allocated MF are retrieved using the association with VNF/PNF and the MF termination request is sent to the radio plugin, The MF termination requested is performed using a new SBI API reported in Table 7.

TABLE 7: 5GR-RL SBI API FOR MF MANAGEMENT

Name	Method	Endpoint	Description
SBI MF Association	GET	/nfs	Retrieve the mapping between MFs and PNFs/VNFs/
SBI MF Instantiation	POST	/mfs	Request instantiation of MF
SBI MF Termination	DELETE	/mfs	Request termination of MF previously instantiated.

3.1.1.4 Innovation contribution to cover gaps

RL PNF Integration

This innovation adds a new feature to the 5Gr-RL, which enables the management of physical radio devices like antennae or baseband units. Specifically, the functions executed by the physical radio device are represented using the PNF information model defined by ETSI NFV. Each domain under the 5Gr-RL control provides the PNF capability that are reported in the abstract view and PNF are orchestrated using the new NBI and SBI API (see "5Gr-RL enhancement" above for details).

RL geo-specific resources

5Gr-RL provides the geographical information for VNF/PNF management in the abstract view and let the corresponding plugin to allocate/terminate resources that are located in a specific geographical area (geo-specific resources). This innovation enables 5Gr-RL to explicitly request (via SBI API) the allocation/termination of specific geo-specific resources.

RAN support

This innovation adds a new feature to 5Gr-RL enabling the configuration of physical radio devices (like antennae, baseband units). It complements the VNF/PNF orchestration features by configuring the radio devices to make it work for the requested network slice. As reported in detail above, for each VNF/PNF a set of specific MFs are defined and applied when the VNF/PNF is used for network slice.

3.1.2 Innovation 2 (I2): Support of Verticals. Vertical-service monitoring

The 5Gr-VoMS is responsible for the monitoring and logging aggregation for the 5Growth platform and has vertical-service monitoring extensions to support vertical-specific use-cases. These are methods developed for the collection of application/service metrics or logs. Such approaches allow evaluating service QoS and bring much more capabilities for the verticals to control and operate with their workloads, established on the 5Growth platform.

For the collection of application metrics, there are two methods that can be applied. The first method is to collect application metrics from the client side. It is available through the instantiation of client-side probes for gathering the monitoring data exactly from the the vertical application or server/VM which runs it. Client-side probes installation depends on the software platform. The client-side probe can be:

- built-in the vertical's application web page and it starts automatically when the client opens it (instantiation of such client-side probe should be considered by the vertical application developers);
- built-in application that provides a service. (instantiation of such client-side probe should be considered by the vertical application developers as well);

- can be a separated software application collocated with the vertical application (instantiation of such client-side probe should be considered as a part of the vertical application installation automation). The usage of client-side probes is shown in Figure 12.

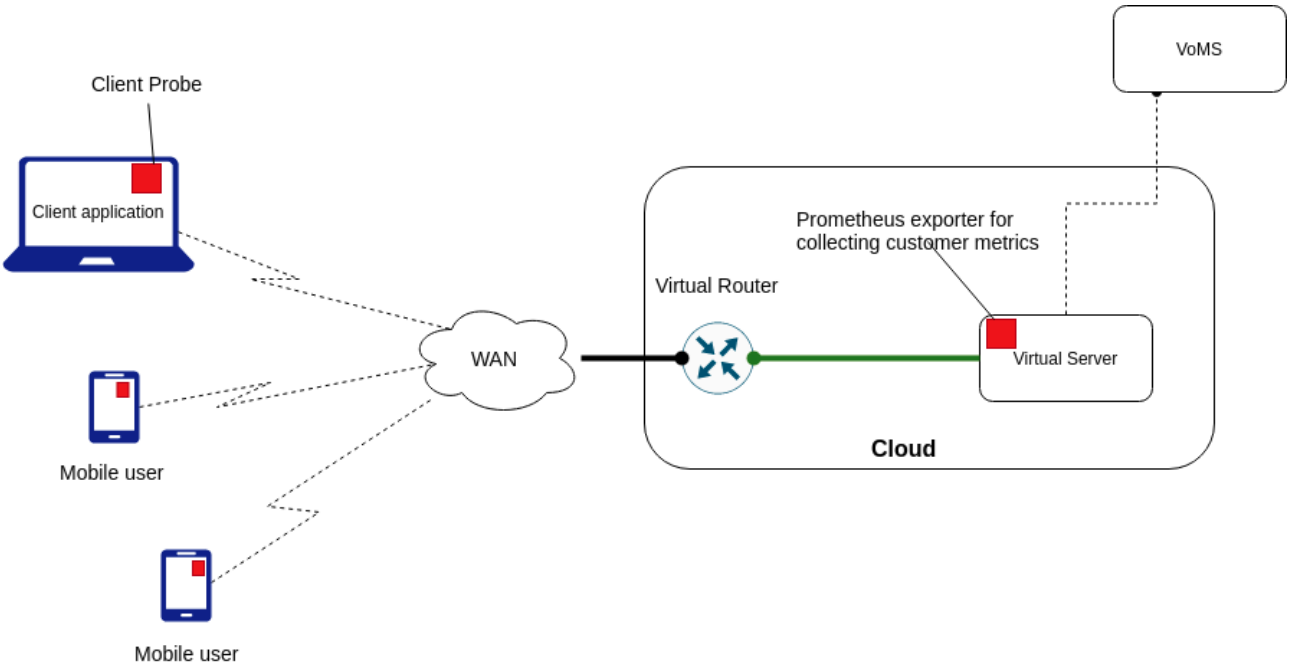


FIGURE 12: CLIENT-SIDE PROBE SCHEMA

An example deployment with a virtual server is shown on the right side of Figure 12. The virtual server has a Prometheus exporter for collecting customers'/clients' metrics (PECCM). This exporter receives metrics from clients and saves them in memory. The Monitoring Platform grabs the collected metrics from the exporter by a request which are afterward processed. PECCM is installed dynamically according to NSD by using the RVM agent.

The client probe can work in two modes: as third-party client-side monitoring and as native client-side monitoring.

In the third-party client-side monitoring mode, the client probe sends a request, receives a reply and calculates the latency time. Then it sends a request to the Prometheus exporter with the information about the latency time. The current latency measuring method shows the indirect quality of the client-side service. It gives the possibility to measure the quality of a network connection. Client probe can send information about the client hardware: CPU usage, memory usage, SNR level, etc. The amount and type of the information depend on the software and hardware platform where the client application is implemented. This mode has the disadvantage that it uses extra network bandwidth.

Native client-side monitoring mode is the mode where the client probe gets metrics from a client application such as latency, speed of data loading, buffer usage, data processing time on the client side, etc. The request latency, in this case, shows the actual QoS. The speed of data loading shows

the quality of service for the data channel. Data processing time on the client-side shows the performance of client hardware if a client has enough performance to use the service.

This mode requires the client probe to pass application metrics. The client probe sends collected metrics to the Prometheus exporter for collecting customer metrics. The monitoring platform receives metrics from the Prometheus exporter as a reply to the request. This mode does not use extra network bandwidth.

Below you can see example of the application metrics of a service:

latency_ms{client_host="192.168.122.1",session_id="1839"} 66.4

loading_ms{client_host="192.168.122.1",session_id="1839"} 146.4

parsing_ms{client_host="192.168.122.1",session_id="1839"} 49.9

bw_kbps{client_host="192.168.122.1",session_id="1839"} 30805.8

where:

client_host is the IP address of the client who uses a service.

session_id is an identifier of a session.

latency is a time interval between the moment client's application requests the data and when it receives the first packet of the data.

loading is a time interval between the moment when client's application receives the first packet of the data and when it receives the last packet of the data.

parsing is a time interval during which client's application processes the data.

bw_kbps shows the bandwidth used when downloading the data, unit of measurement is Kbps.

The timing diagram of data processing is shown below in Figure 13:

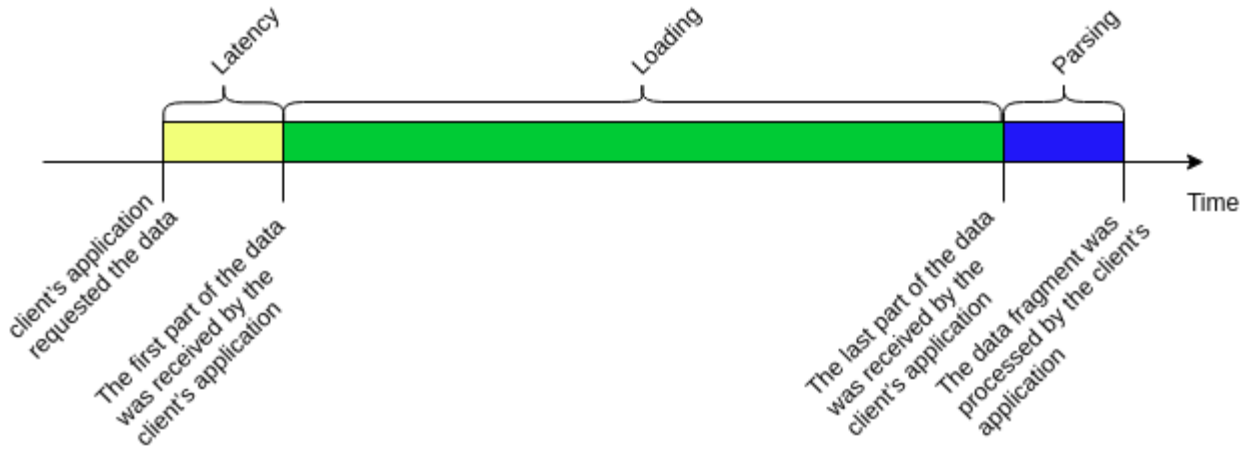


FIGURE 13: TIME DIAGRAM OF DATA PROCESSING

The second method for gathering application metrics is collecting application metrics from the server-side. The Server-side probe is a dedicated server (instance or VM in this particular case) that makes requests to the virtual server as a client and reports to the monitoring platform about service status. The server-side probe gives the possibility to evaluate the quality of the service without network channel influence. Metrics from this server are collected by the monitoring platform. The dedicated server must be defined in the NSD. The simple service checks can be unified for example checking application protocols availability (HTTP, HTTP, RTP, FTP, etc..), but these simple checks do not give the possibility to fully check the service that users receive. A server-side probe can be tied to a specific service type, and service developers should develop the service checking procedure. The server-side probe is shown in the Figure 14. Service-side can have the same functions and modes as the client-side probe.

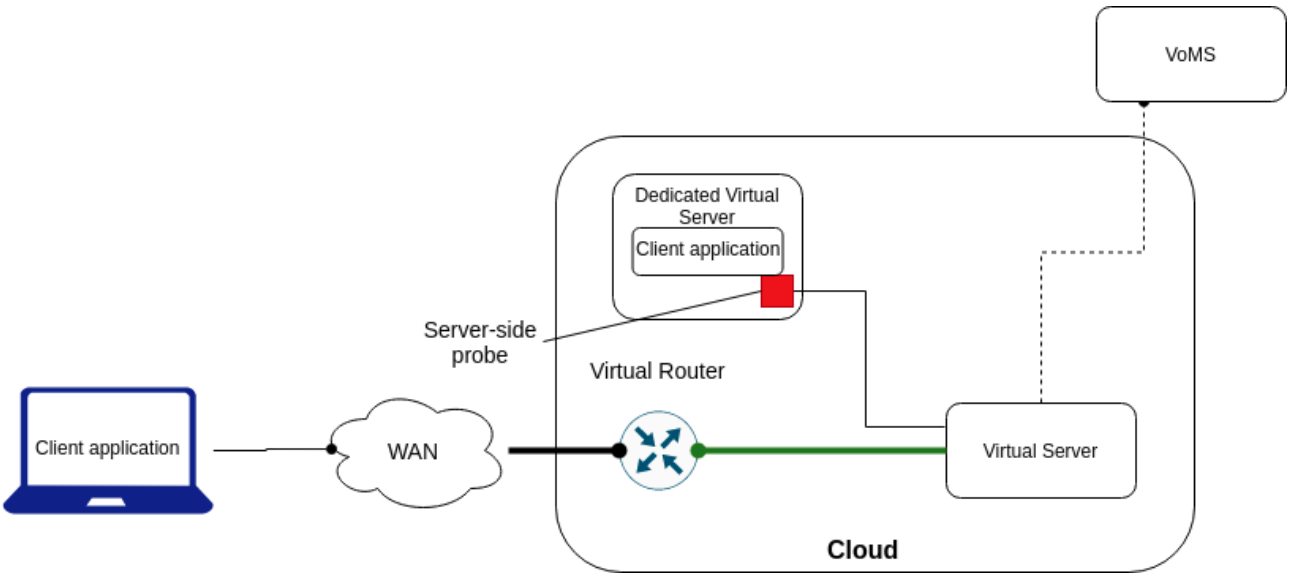


FIGURE 14: SERVER-SIDE PROBE SCHEMA

The method of gathering logs from VNFs implies the deployment of Filebeat in the targeted VNFs through the RVM Agent, including the proper Filebeat configuration for extracting the data from a given log file and publish the data in the correct Kafka topic, as explained in Figure 7. The Filebeat configuration file format would follow this format:

```
filebeat.inputs:
- type: log
  paths:
    {% for path in paths -%}
    - {{path}}
    {% endfor %}

output.kafka:
  hosts: ["{{kafka_bootstrap_server}}"]
  topic: "{{nsId}}"
```

As a result, when logs are generated, Filebeat publishes each file line in Kafka in a specific message, formatted in JSON, which is eventually processed by the Logs block in the 5Gr-VoMS platform.

The use cases enabled by the integration of a log monitoring system are the following:

- Resource management: log data streams priorization based on severity level, monitoring across systems to detect particular log events and patterns in log data, real-time monitoring for anomalies or inactivity to gauge system health, identify performance/config issues.
- Application troubleshooting: assessing application health, diagnosis and identification of the root cause of errors.
- Performance improvement: detect bottlenecks, optimize processes, discover hard-to-find bugs.

3.1.2.1 Innovation contribution to cover gaps

VS arbitration at runtime

This innovation enables the 5Gr-VoMS to be dynamically configured to retrieve the vertical service metrics that determine the dimensions of the network slice associated with the vertical services and metrics that could be used to dynamically trigger priority-based vertical service lifecycle management procedures (i.e., termination or scaling). In this sense, this innovation provides enhanced support for vertical service arbitration at runtime. In the simplest scenario, an alert-based mechanism could be configured on the 5Gr-VoMs to trigger the vertical service action (i.e., scaling or termination) once a certain threshold is reached. In a more advanced scenario, another component leveraging AI/ML algorithms could be used to dynamically scale or terminate the vertical services based on the resource availability and established vertical service priorities.

SO automatic network services management

One of the key features of the 5Gr-VoMS is configurability through an API, which allows interacting easily with other 5Growth components. Every component of the 5Growth stack (whether 5Gr-VS, 5Gr-SO, or 5Gr-RL) can configure 5Gr-VoMS monitoring jobs according to their requirements. The configured monitoring jobs collect requested metrics from different sources on any architecture layer (resources, services, vertical services) or logs from specific VNFs and generate alerts. The SO automatic network services management is enabled by the 5Gr-VoMS capability of collecting metrics and logs from vertical's service VNFs. This monitoring data is crucial for monitoring the usage level (network throughput, CPU, RAM, disk usage, etc.) of the vertical's service. According to this data, 5Gr-SO can trigger a scaling process (scaling in or out) for the vertical application in an automated way. The 5Gr-VoMS is also able to track vertical service health (reachability from a certain location or response time) and notify 5Gr-SO or 5Gr-VS by raising an alert, in case a certain threshold is exceeded. In such cases, the SO and VS services can decide whether to scale or redeploy the application to perform auto-healing. Services monitoring ways and approaches are already described in Section 3.1.2 and the interaction options are shown in Figure 15.

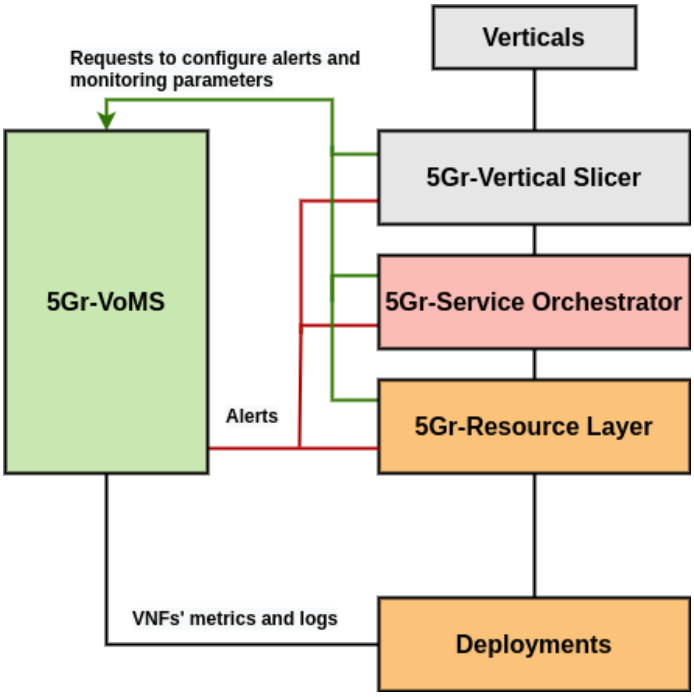


FIGURE 15: 5GR-VOMS USAGE IN 5GROWTH ARCHITECTURE

This feature is covered according to the way the Monitoring Platform has been built for both metric and log monitoring systems.

SO self-adaptation actions

Originally, the 5Gr-SO had the option to configure statically defined thresholds and actions for alerts. The 5Gr-SO analysed Network Service Descriptor (NSD) and made requests to 5Gr-VoMS for configuration monitoring parameters, alerts, and actions in case of alerts activating. Alerts were defined in NSD and used for VNF scaling. The AI/ML platform integration into the 5Growth stack turned this process to self-adaptive. D2.2 [3] describes the usage of 5Gr-AI/ML when interacting with the 5Gr-SO. The Network Service Descriptor (NSD) was extended to support the AI/ML-driven scaling operation and 5Gr-AIMLP was integrated into 5Growth stack. The 5Gr-AIMLP is based on Apache Spark.

As it was mentioned, the NSD was extended with AI/ML configuration information. 5Gr-SO analyses the NSD and makes requests for monitoring parameters configuration and configuration of visualization dashboards in 5Gr-VoMS. Then it makes the request to Kafka MQ for creating Kafka Topic with the specific name.. Namely, the 5Gr-SO contacts with the 5Gr-VoMS to create/delete "data spaces" (named topics) in Apache Kafka to insert the required monitoring data for AI/ML-driven scaling operation; and to create/delete "data scrapers" that filters the required monitoring data out of all available monitoring data. "Data scrapers" are elements that filter out the monitoring data available in the 5Gr-VoMS and insert them in the requested Kafka topic for the Apache spark which process checking continuously the performance of the NFV-NS against the AI/ML module to detect SLA deviations and trigger the corrective actions. Also, the 5Gr-SO holds AI/ML models, loads the necessary AI/ML model from the 5Gr-AIMLP according to NSD and launch the Apache Spark jobs in

charge of checking the SLA compliance. These AI/ML models and processing routines are requested for the AI/ML platform. AI/ML techniques allow performing real-time data analytics to detecting anomalies, forecasting, and then making decisions about actions for optimization and reconfiguration of the service and/or the system (such as auto-scaling, self-healing, etc.). The same methodology can be applied to other layers, like the 5Gr-RL to predict and react in case of problems in the infrastructure layer.

SO dynamic monitoring orchestration

For each vertical service making use of the 5Growth platform, the Monitoring Platform applies a full control of the lifecycle of the monitor functions used to collect the metrics and logs, enabling the orchestration of these processes to automatically create/delete the desired configuration in the targeted scenario and within the components that belong to the 5Gr-VoMS.

3.1.3 Innovation 3 (I3): Monitoring Orchestration

The Monitoring Orchestration innovation intends to provide automation capabilities to the 5Gr-VoMS, achieving the integration of features related to the full control of the lifecycle of the monitor functions instances that compose the Monitoring Platform, among other capabilities that will be reviewed in this section.

To illustrate the first topic aforementioned, the following example will describe the installation of a set of monitoring instances requested by the 5Gr-SO. The process description will start when the 5Gr-SO receives the request to instantiate the service. The 5Gr-SO has the Monitoring Manager unit which is responsible for interacting with 5Gr-VoMS. Also, it is responsible for analysing the "monitoredInfo" section of NSD which holds information about all tracked parameters. An example of a monitoring parameter from the NSD is the following:

```
[
  {
  "monitoringParameter": {
    "monitoringParameterId": "mp9",
    "name": "onewaylatency",
    "performanceMetric": "linklatency.spr2.webserver",
    "type": "link_metric",
    "params": {
      "ip": "vnf.webserver.vdu.webserver_vdu.intcp.webDistExt.address",
      "polling": 2
    },
   "exporter": "onewaylatency_exporter"
  }
]
```

Where:

monitoringParameterId - is the unique identifier of this monitoring parameter.

name - is a human-readable name of the monitoring parameter.

performanceMetric - is a parameter that defines the metric's name and the name of the VNF on which metrics are collected. Also, this parameter can be extended: it can have a metric name, a source and a destination name VNFs as it was shown in this example.

params - is a dictionary structure of parameters for exporter.

ip - is an IFA reference to the IP address of the VNF to which latency measurements will be done.

polling - is a period of how often latency will be evaluated, unit seconds.

target - is a reference for the probe.

module - is a type of the probe exporter which provides metrics.

type - is a type of monitoring parameter. It can have, as value:

general (default) - this type is used for measuring hardware metrics.

link_metric - this type is used for link's metrics between two VNFs.

application_metric - this type is used for measuring application metrics.

logs - this type is used for log collection exporters (monitoring probe).

According to the above monitoring parameter, the Monitoring Manager sends the request to 5Gr-VoMS for installation "onewaylatency_exporter" at "spr2" and "webserver" VNFs by using RVM agents. The "spr2" VNF initiates measurement requests to "webserver" VNF. 5Gr-VoMS configures the Prometheus collector at RVM agent of "spr2" by using fields from the "params" section. The Prometheus collector sends requests to the exporter (monitoring probe) every N second and gets metrics from it. Then the Prometheus collector publishes these metrics to the specific Kafka topic from which the Prometheus PushGateway gathers metrics and provides them to the Prometheus server. The same approach is used for monitoring parameters with "application_metric" type, but in this case a dedicated server is used for service monitoring. This dedicated server should be defined in NSD. The exporter (monitoring probe) is installed only at the source VNF using application_metric type.

The example of monitoring parameter with **logs** type is the following:

```
{
  "monitoringParameter": {
  "monitoringParameterId": "mp12",
  "name": "logs",
  "performanceMetric": "logs.spr2",
  "type": "logs",
  "params": {
    "file1": "/var/log/syslog"
    },
  "exporter": "filebeat"
 }
}
```

This example of the monitoring parameter has the same fields as the previous example. The difference here is the Monitoring Manager sends the request to 5Gr-VoMS for installing exporter (monitoring probe) at "spr2" VNF. 5Gr-VoMS sends the request for installing Filebeat and its configuration on "spr2" VNF by using the RVM Agent. Prometheus collector is not used in this case. The Filebeat publishes logs from tracked file "/var/log/syslog" to a specific Kafka topic which is equal to the NS identifier. 5Gr-VoMS creates a configuration file for Filebeat based on a template and the name of the Kafka topic for logs configures at that moment.

Apart from that workflow, The 5Gr-VoMS also configures the Log Monitoring pipeline. For doing this, the Config Manager sends the proper requests to the Log Pipeline Manager to properly configure all the entities involved in the Log Monitoring pipeline (e.g., the Elastic Stack, the Kafka topic to be used by Filebeat and Logstash).

And finally, the last example of monitoring parameter with type general (default) is used for measuring hardware metrics. The example is following:

```
{
  "monitoringParameter": {
  "monitoringParameterId": "mp8",
  "name": "webserverLoad",
  "performanceMetric": "ByteIncoming.webserver.eth0",
  "exporter": "node_exporter",
  "type": "general",
  }
}
```

The parameter "performanceMetric" is split into three parts: metric's name, name of the VNF on which metrics are collected, and the name of the interface inside the VNF which generates this metric. The exporter will be installed at "webserver" VNF and the Prometheus collector will be configured for gathering metrics from the exporter and publishing it into the Kafka topic.

Regarding other features related to the Monitoring Orchestration innovation, the inclusion of the RVM Agent allows the fulfilment of the functionality related to the dynamic deployment of monitoring probes, allowing the orchestrator to auto-discover relevant probes, leveraging in the analysis and post-processing of monitoring probes metadata gathered from the infrastructure.

3.1.3.1 Innovation contribution to cover gaps

VS arbitration at runtime

The extraction of monitoring information from vertical service is critical for the training and execution of the AI/ML based algorithms used in the control-loops, and this innovation enables to install on-demand the agents and probes which enable this extraction. Moreover, it allows to modify the metrics extracted from the vertical service over time, in order to use every time more evolved control-loops algorithms (such as the AI/ML based arbitration explained in I5 in Section 3.1.5).

SO automatic network services management

The 5Growth platform is designed to simultaneously run and operate with many vertical services or applications. In such conditions, it is not acceptable to configure and create monitoring settings manually for each instance. The monitoring orchestration is the ability of the 5Gr-VoMS to automatically create set up and deprovision of the monitoring instances (jobs, probes, etc,). As it is described in Section 2.5, 5Gr-VoMS has the option to receive requests from any block of the 5Growth stack. The automated requests, received from the 5Growth services are enabling the 5Gr-VoMS monitoring orchestration for track monitoring parameters and alerts.

SO dynamic monitoring orchestration

According to the way the Monitoring Platform has been built for both metric and log monitoring systems, the objective of controlling the lifecycle of the monitor functions instances that compose the Monitoring Platform is completely fulfilled, as both the metrics and logs blocks are automated by the Config Manager and the Log Pipeline Manager respectively.

3.1.4 Innovation 4 (I4): Control and Management. Control-loops stability

This section presents the innovations introduced in the 5Growth architecture and the workflows performing automated close-loop process handling: (i) the automated update of arbitration policies at the vertical slicer (5Gr-VS) level for automated slicing optimization; (ii) and the automated scaling of network services at the service orchestrator (5Gr-SO) level; leveraging on the AI services provided by the 5Gr-AIMLP developed in I5 (see Section 3.1.5). The integration of AI features into the 5Growth architecture follows the O-RAN architectural concepts explained in [40] where there is a split between training and model execution. Specifically, the 5Gr-AIMLP performs the model training and provides trained-model to other building blocks of the 5Gr architecture, which performs the inference [41]. A proof-of-concept validation of the automated network service scaling at the 5Gr-SO level is described in Section 4.3.

3.1.4.1 5Gr-VS level closed-loop control

The evolution of the last version of the 5Gr-VS has introduced mechanisms to move from static arbitration algorithms to a data-driven arbitration approach, supported by the 5Gr-AIMLP, which allows to automatically regulate the arbitration decisions based on the service level context dynamicity. In particular, the arbitrator embedded internal static and predefined algorithms, available on the first release of the 5Gr-VS, have been extended to support the configuration of arbitration policies which enable to automatically retrieve trained models from the 5Gr-AIMLP platform, in order to be used to determine the arbitration decisions during the vertical service instantiation and runtime phases. These arbitration decisions determin if a lower priority vertical service instance is to be terminated to release resources in order to accommodate the incoming or scaled vertical service, and if existing network slice instances or network slice subnet instances can be shared.

The arbitration policies are onboarded on the 5Gr-VS, by an Admin user [102], and establish when to retrieve a trained model (e.g., for each arbitration decision, after a certain period of time), and to which vertical service requests they should be applied (e.g., for which tenant, VSB or slice service type). A representation of this workflow is shown in Figure 16. Closed-loop automation is enabled by the continuous re-training of the AI/ML models, which in this implementation are based on service and slice level information coming from the 5Gr-VS (e.g., number of vertical service and slice instances, slice service types, network slice placement constraints), and/or vertical application information coming from the 5Gr-VoMS. This loop implements the chain of monitoring-analyse-decide-actuate actions for automated slicing optimization. The processing of data coming only from the 5Gr-VS architectural layer (related to vertical services and network slices only) and from the vertical application itself allows to keep the functional separation between the 5Growth architectural components and it is particularly suitable for scenarios where 5Gr-VS and 5Gr-SO are managed by different actors. An extended version of this approach could also involve the processing of data coming also from the 5Gr-SO (e.g., network service and resources) to train the models enabling cross-layer decisions through the exchange and the joint elaboration of monitoring data related to different layers.

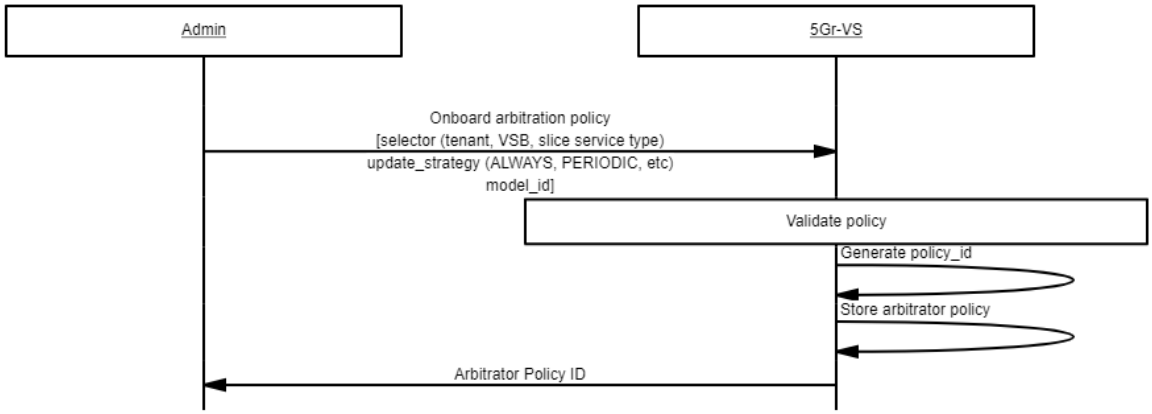


FIGURE 16: 5GR-VS ARBITRATOR POLICY ONBOARDING WORKFLOW

During the vertical service instantiation and operation phases, the 5Gr-VS determines the policy to be used for the arbitration model, and if it needs to retrieve a trained model from the 5Gr-AIMLP, as shown in Figure 17.

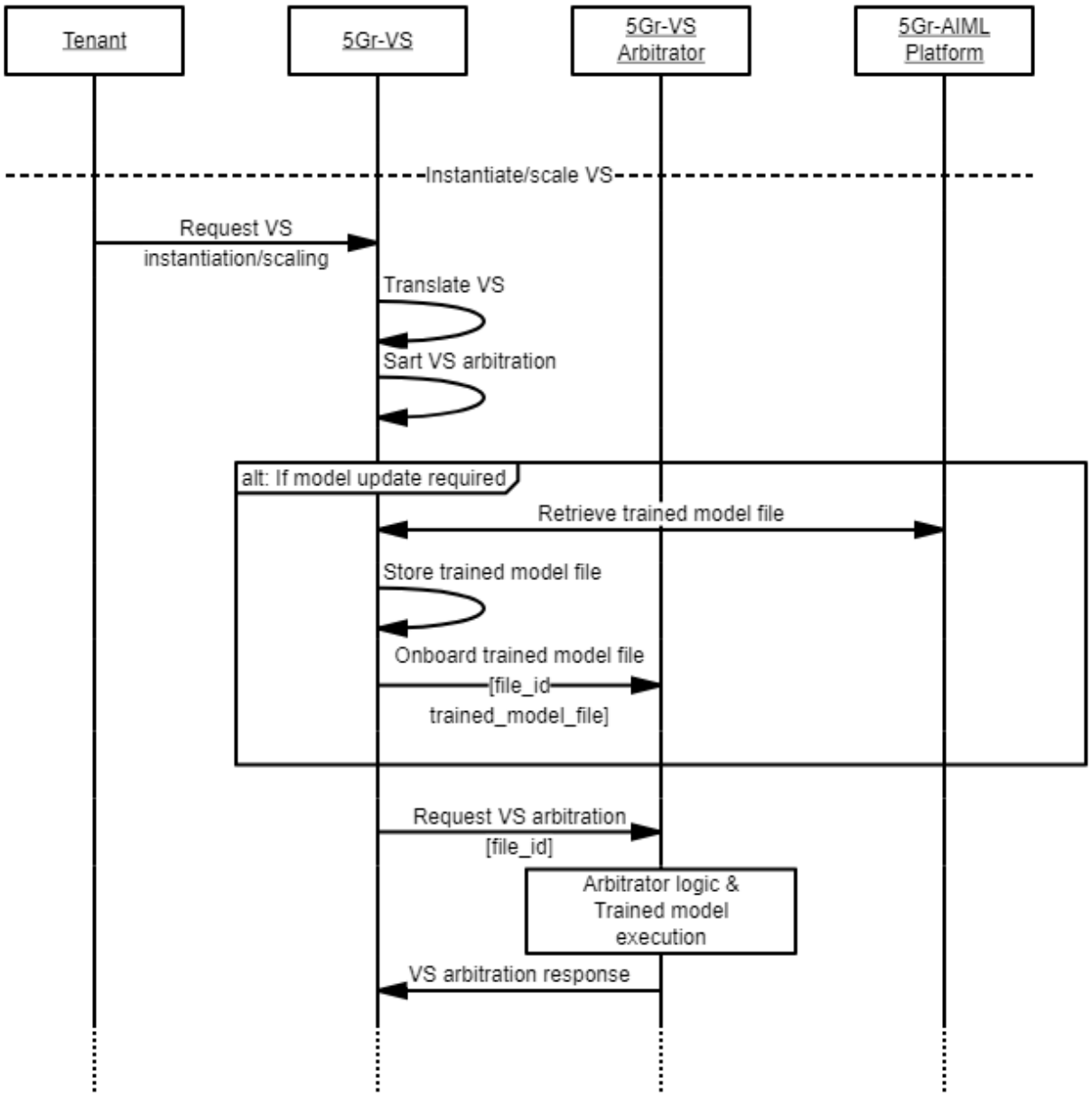


FIGURE 17: 5GR-VS ARBITRATION TRAINED MODEL UPDATE

3.1.4.2 5Gr-SO level closed-loop control

Before starting with the description of the architectural enhancements of the 5Gr-SO, we proceed to describe the new AI/ML information element (IE) extending the ETSI NFV-IFA 014 [6] NSD template that triggers the process. An example of this AI/ML IE is the following:

```
      "aimlRules": [
            { "ruleId": "aiml_rule1",
              "problem": "scaling",
              "nsMonitoringParamRef": ["mp1", "mp2"]
            }
                  ]
```

This new IE expresses the need of interaction with the 5Gr-AIMLP to configure AI/ML-based decisions for a given MANO problem ("scaling") and specifies the metrics out of the ones already defined for this kind of network service in the NSD field *"monitoredInfo"* required by this AI/ML problem (*"mp1"* and *"mp2"*) to perform its decisions.

Figure 18 presents the 5Growth stack with a focus on the architecture of the 5Gr-SO. The marked submodules are the ones that have been modified/added with respect to the baseline architecture (that of 5G-TRANSFORMER project [42]) to support the closed-loop innovation on the AI/ML-based scaling operations:

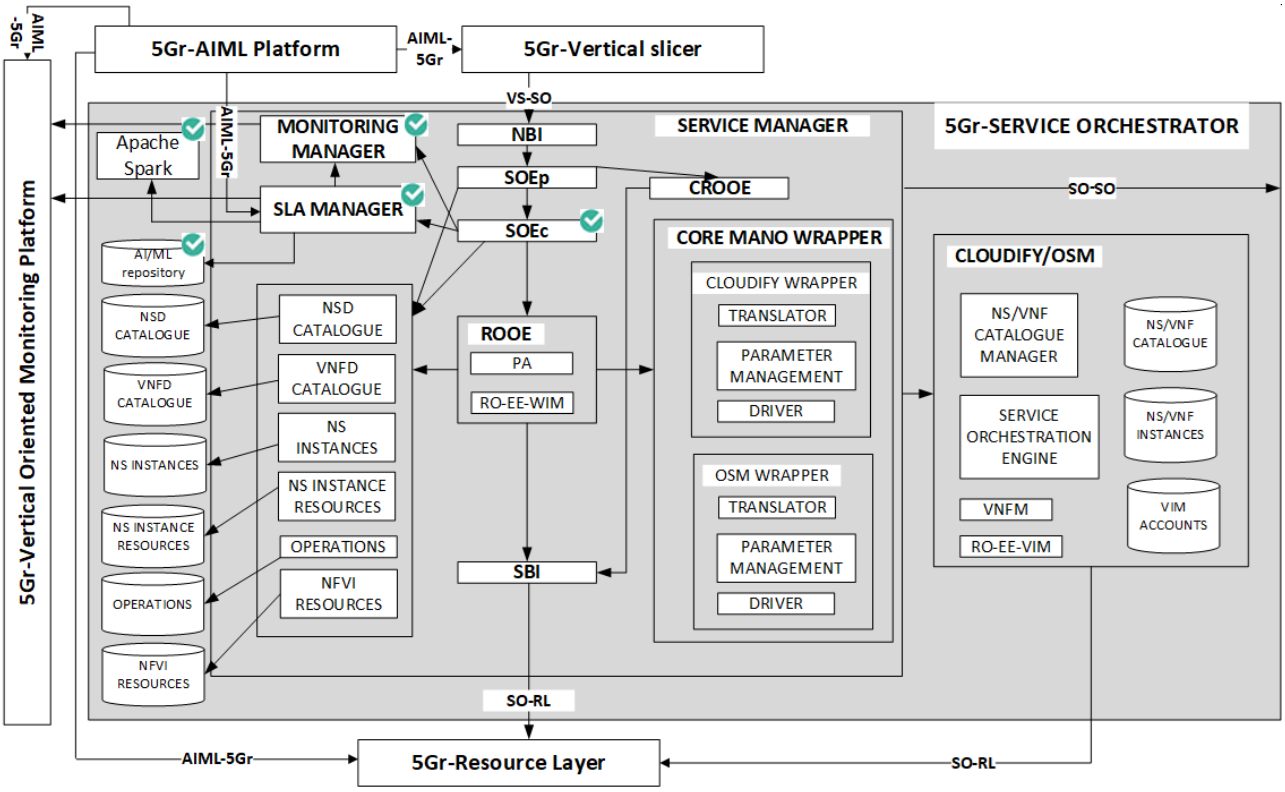


FIGURE 18: 5GR-SO ARCHITECTURE EXTENSION TO SUPPORT AI/ML-BASED SCALING OPERATIONS

SLA Manager: This submodule is responsible for handling the NFV-NS SLA compliance and triggering the scaling process in case of SLA violation. Initially, the detection was based upon an alerting system configured through the 5Gr-VoMS. Now, it has been extended to orchestrate all the operations to handle the AI/ML-based scaling operation when including the described IE in the NSD. For this purpose, it interacts mainly with the Monitoring Manager submodule, the 5Gr-AIMLP, and with Apache Spark [99]. The inference operation is done in the 5Gr-SO, using Apache Spark and orchestrated by the SLA Manager.

Monitoring Manager: This submodule interacts with the 5Gr-VoMS to configure the collection of performance metrics expressed in the NSD and configuration of visualization dashboards. Now, its interaction with the 5Gr-VoMS has been extended to configure dedicated Apache Kafka topics [100]. For this innovation, the monitoring data expressed in the AI/ML IE of the NSD is inserted into the Kafka topics to be processed by the Apache Spark inference job.

Service Orchestrator Engine child (SOEc) this submodule of the Service Orchestrator Engine (SOE), handling the orchestration of regular (i.e., non-composite) NFV-NSs, has been extended to interact with the new capabilities of the SLA Manager module and trigger the configuration of the close-loop operations.

Furthermore, the 5Gr-SO architecture includes an instance of **Apache Spark** and the **AI/ML repository**, which is a new submodule where the SLA Manager stores the requested AI/ML models and processing routines obtained from the 5Gr-AIMLP. These models and processing routines are required to launch the Apache Spark jobs in charge of checking the SLA compliance and deciding on the appropriate instantiation level of the NFV-NS for the given service demands.

Finally, the **5Gr-VoMS** has also been extended to support the AI/ML-based scaling operation. In addition, to host the Apache Kafka platform, the 5Gr-VoMS adds a REST-API to create *data scraper* elements. These elements, upon the request orchestrated by the SLA Manager, filter out the collected monitoring data for an NFV-NS by the Prometheus platform used in the 5Gr-VoMS and insert them in the requested Kafka topic to be ingested by the Apache Spark streaming processes performing inference operations.

Figure 19 presents the workflow followed by the 5Gr-SO to configure the AI/ML-based scaling operation [41]. This workflow takes as starting point the last step of the instantiation process (after VNFs have been allocated and their interconnections and monitoring jobs have been configured), when the SOEc contacts the SLA Manager.

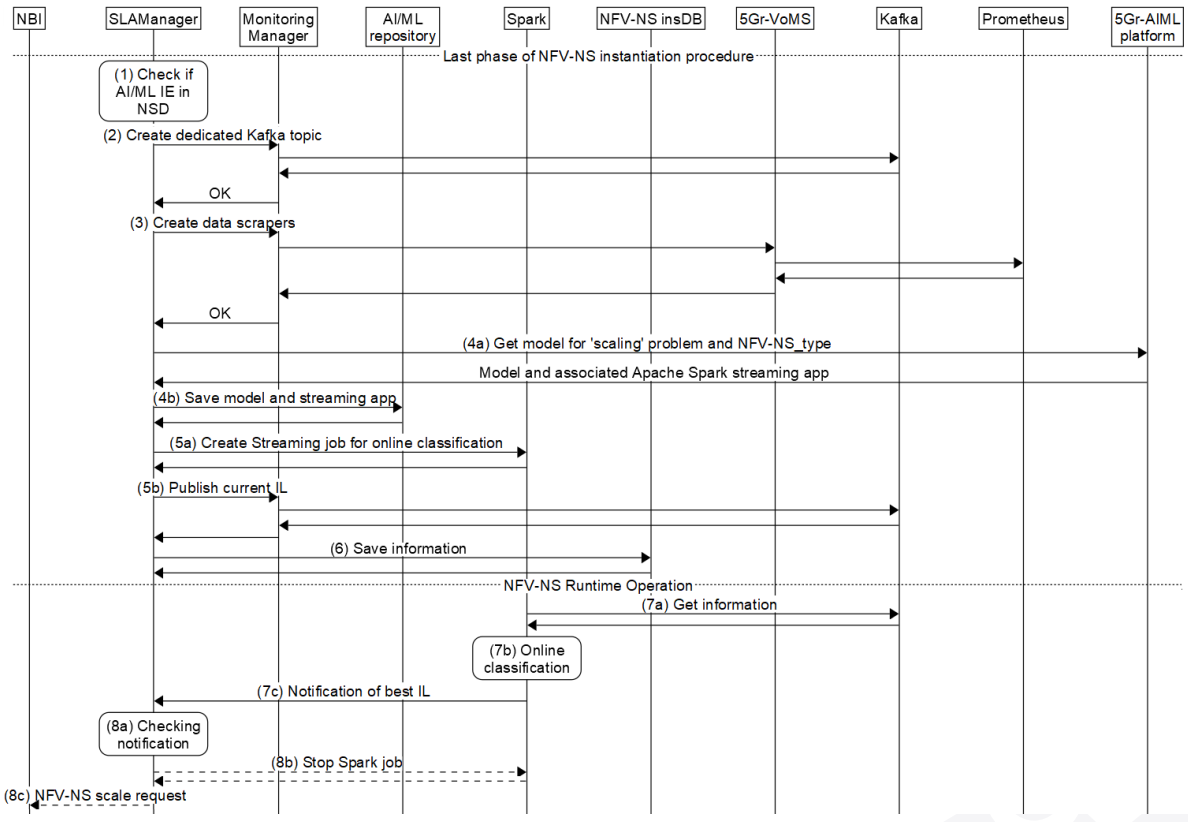


FIGURE 19: CLOSED-LOOP WORKFLOW FOR THE AI/ML-BASED SCALING OPERATION

The workflow is as follows:

1. The SLA Manager checks the existence of an AI/ML IE in the NSD for a scaling problem. The next steps happen upon a positive confirmation.
2. The SLA Manager contacts the Monitoring Manager to configure a dedicated data topic in the Apache Kafka instance run by the 5Gr-VoMS to insert the required monitoring information expressed in the NSD in order to handle the AI/ML-based scaling operation.
3. The SLA Manager, through the Monitoring Manager, creates the "data scrapers" elements at the 5Gr-VoMS to filter out the monitoring data specified at the AI/ML IE.
4. The SLA Manager would contact the 5Gr-AIMLP (see I5 in Section 3.1.5) to download the required model and its associated streaming application to perform online classification and stores them in the AI/ML repository.
5. The SLA Manager launches the Apache Spark streaming job. In this work, the SLA manager publishes the current NFV-NS instantiation level (IL) in the dedicated Apache Kafka topic created in step 2) to give the appropriate context to the Apache Spark application.
6. Finally, the SLA Manager saves the AI/ML-based information (Apache Kafka topic, data scrapper, Apache Spark job references) in the NFV-NS instance database.
7. From this point on, periodically, the Apache Spark job ingests the data requested in step 3) from the Kafka topic, performs online classification, and notifies the result (i.e., the best IL given the current context) to the SLA Manager.
8. The SLA Manager checks the notification, and if the received IL differs from the current IL, it stops the Apache Spark Job and triggers the scaling operation through the northbound interface of the 5Gr-SO.

In case of scaling, the 5Gr-SO proceeds to create/terminate the required VNFs instances as indicated by the new IL, updating the interconnections between VNFs and the performance monitoring jobs accordingly. As the last step of the scaling procedure, added by the integration of AI/ML-based operations, the SOEc contacts the SLA Manager, which, when retrieving the information from the NFV-NS instance database, repeats steps from 5) to 7) to close the loop.

The 5Growth architecture could also accommodate other types of AI/ML-based problems specified in the presented IE. For that, we can apply the same general workflow as described above for the interaction with the 5Gr-AIMLP and the streaming functions of Apache Spark..

3.1.4.3 Innovation contribution to cover gaps

Enhanced VS network slice sharing and VS arbitration at runtime

This innovation enabled the usage of enhanced arbitration algorithms that rely on AI/ML models available on the AI/ML platform. In particular the algorithms aimed to improve the overall workflow that determines the network slice instances to be shared among vertical services during the instantiation phase, as further detailed in Section 3.1.5. Future work may also use this innovation to provide algorithms dynamically triggering vertical service lifecycle management events to prevent SLA breaches.

VS dynamic service composition

As detailed in Section 3.1.6, in 5Growth, vertical services can be decomposed in vertical (sub)services and in network slice and network slice subnets. Currently, the vertical (sub)service decomposition is established during the service design phase associating components of the Vertical Service Blueprints (VSB) to vertical (sub)services and to the domain to which these (sub)services should be requested. The network slice and network slice subnet, on the other hand, is currently determined during the instantiation phase by means of translation policies. Future work may use this innovation to enable enhanced translation mechanisms to determine the decomposition of the vertical service. In particular, this innovation could be used in combination with the new northbound API of the 5Gr-VS, which enables a more flexible management of the translation policies, to update the vertical service to network slice mapping using AI/ML based algorithms. A similar approach could also be used to improve the vertical service to vertical (sub)service mappings.

SO automatic network service management

This innovation provides the 5Gr-SO the procedures to interact with different building blocks of the 5Growth including the AI/ML platform (I5), the 5Gr-VoMS (I2), and potentially the forecasting platform (I10). These interactions allow the creation of automated close-loop operations using different ML algorithms, such as the ones designed for the automated service scaling designed in I8 (see Section 3.2.1).

SO self-adaptation actions

Automated SLA management in the form of scaling operations can be performed in the 5Gr-SO module with the introduced enhancements to react in front of dynamic (and potentially foreseen) changes of the network and available resources.

SO dynamic monitoring orchestration

Through the interaction with the Monitoring Platform to properly configure the scrapers to interact with the AI/ML platform, the automation of the configuration of these modules is fulfilled, supporting the orchestration and control of the lifecycle of the data scrapers created within the 5Gr-VoMS. By achieving this, the AI/ML platform can make use of the monitoring data collected by the 5Gr-VoMS for enhancing the closed-loop features. This data is properly gathered thanks to the orchestration capabilities offered by the 5Gr-VoMS.

5Gr-SO geo-location dependent federation

This innovation also enables the 5Gr-SO to perform geo-location dependent federation. Especially the auto-scaling of a network service (NFV-NS) can be extended to its nested NFV-NS on the federate domains (see Section 3.1.6.1.2). Upon detection of an SLA violation in a nested NFV-NS at the consumer domain, the SLA manager of the 5Gr-SO triggers an auto-scaling operation of the nested NFV-NS in the provider domain, and then update the inter-nested connections. If the event happens at the provider domain, then the provider domain proceeds to scale the nested NFV-NS and notifies the 5Gr-SO at the consumer domain that a nested NFV-NS is being scaled. The 5Gr-SO at the consumer domain waits until the nested NFV-NS scaling.

3.1.5 Innovation 5 (I5): Control and Management. AI/ML Support

The 5Gr-AIMLP and its interactions with the 5Gr-VS and the 5Gr-SO have been leveraged to implement ML-driven slice sharing within the 5Gr-VS Arbitrator as well as ML-driven service scaling within the 5Gr-SO. In both cases, an ML model is uploaded, along the corresponding data set, is uploaded onto the 5Gr-AIMLP through the external interface, implementing the workflow depicted in Figure 20.

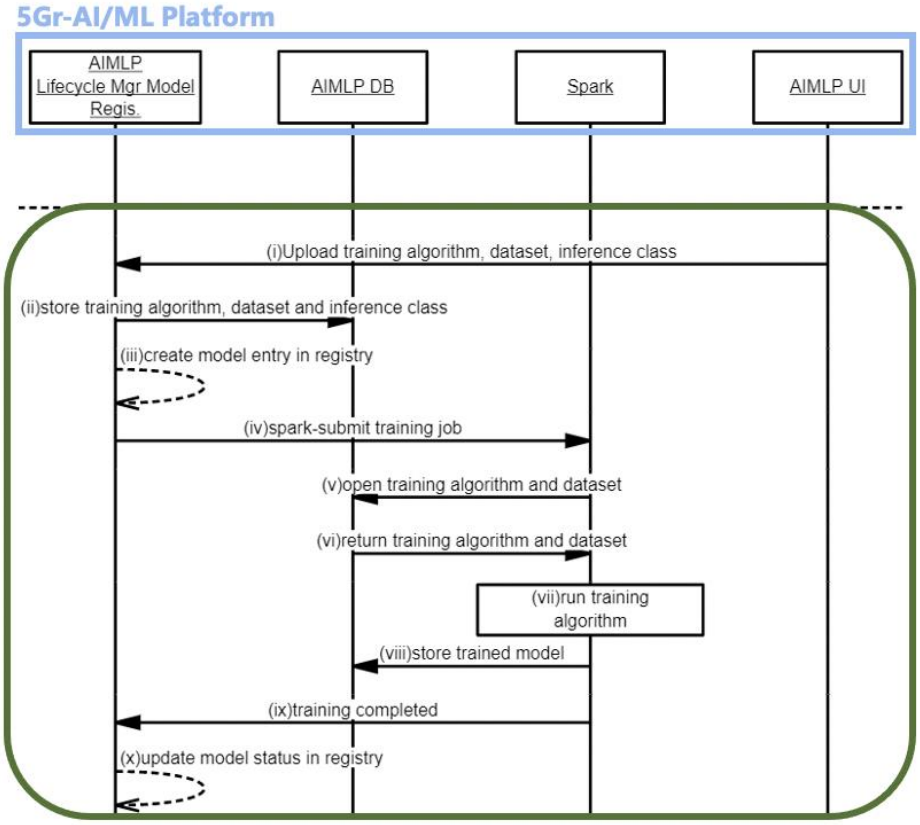


FIGURE 20: ML MODEL ONBOARDING THROUGH THE 5GR-AIMLP

For the ML model onboarding, three files have to be uploaded:

- Training algorithm, which includes the instructions to be executed to train the ML model: dataset reading, translation of features and labels as specified in the algorithm itself, model selection among those available in the Spark/BigDL library, and model parameter setting model training, storage of the trained model. It can be a Python file or JAR (Java Archive);
- Labeled dataset, in csv, json, parquet, or orc format;
- Inference class, needed by the 5Gr-entity (e.g., 5Gr-VS or 5Gr-SO) to perform inference once the trained model is available. Thus, the 5Gr-AIMLP does not use this file. Instead, it includes it along with the trained model into a compressed zip file and delivers it to the 5Gr-entity. It can be a Python file or JAR (Java Archive), and includes the following instructions: acquisition of the data for inference, which are provided as input to the trained model, and inference. As an example, in the case of ML-driven scaling at the 5Gr-SO, the input data is provided by the

5Gr-VoMS. The input data will have to be provided in exactly the same format as that included in the data set and used for the model training.

Upon onboarding the ML model, the external user has also to specify:

- The name of the model.
- The network service descriptor identifier (nsd_id), or the network service category for which the model can be used, e.g., digital twin.
- The scope, i.e., the type of decision-making process for which the model can be used (such as scaling, slice sharing, or forecasting).
- The ML engine, e.g., Spark or BigDL.

The 5Gr-AIMLP adds a timestamp to record the time instant at which the model was onboarded. Then, the model is submitted to the Hadoop cluster and trained through either Spark or BigDL, as specified by the external user. Specifically, the training algorithm connects to the Hadoop distributed file system, retrieves the uploaded dataset and uses the proper ML library to select the model. Upon completion of the ML model training, the output file is stored in the Hadoop distributed file system. The 5Gr-AIMLP then creates a zip file that includes the trained model and the associated inference class, and it generates an URL including the path that a 5Gr-entity requesting the trained model will use to retrieve the necessary files. The ML Lifecycle Manager then monitors its status: it can trigger a new training job either periodically, or whenever new data are available from the 5Gr-VoMS.

Models stored in the 5Gr-AIMLP can be requested by a 5Gr-entity through a REST interface, specifying scope and nsd_id. The 5Gr-AIMLP then provides the aforementioned URL, as also detailed in the example workflow in Figure 21 referring to the interaction between the 5Gr-AIMLP and the 5Gr-SO.

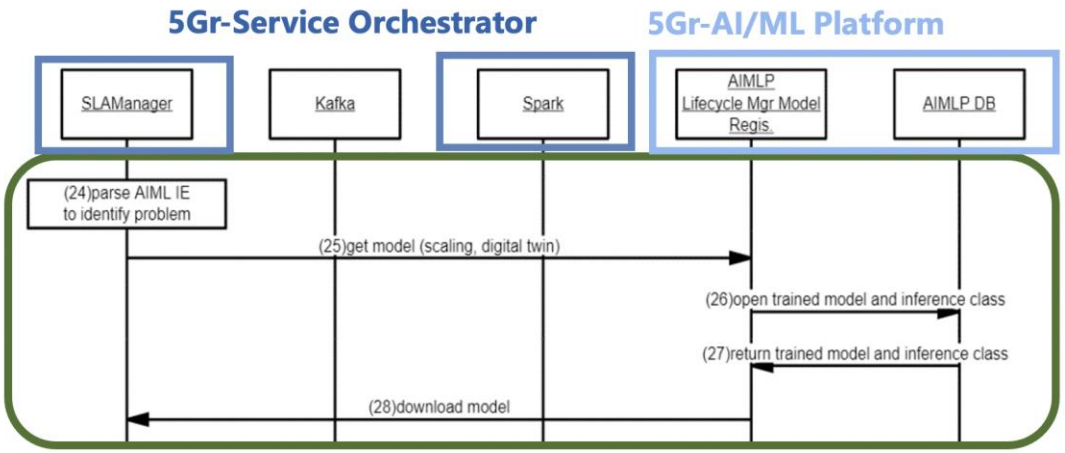


FIGURE 21: WORKFLOW OF THE INTERACTION BETWEEN THE 5GR-AIMLP AND THE 5GR-SO (THE MESSAGE NUMBERING ACCOUNTS FOR THE PREVIOUS STEPS RELATED TO SERVICE INSTANTIATION)

3.1.5.1 Innovation contribution to cover gaps

The AI/ML platform, in charge of model lifecycle management (including model uploading, catalogue building, and model training), realizes the concept of AIMLaaS within the 5Growth architecture. As

detailed below, this allows for the exploitation of AI/ML models for the various decision-making processes necessary in a 5G management and orchestration stack, thus fulfilling the need for fully automated service and network management.

Enhanced VS network slice sharing

This innovation allows the 5Gr-VS and, in particular the Arbitrator, to perform effective slice sharing, by adapting the parameters used by a slice sharing algorithm to the number of service instance requests that the 5Gr-VS receives from the verticals and the impact that they would have on the usage of computing resources. Specifically, through a ML model trained via BigDL and delivered by the 5Gr-AIMLP to the Arbitrator, the latter can receive the best "sharing level" to be used, thus driving the identification of the sharable slices (and sub-slices) through an ML approach.

VS arbitration at runtime

Thanks to the ability of the 5Gr-AIMLP to collect real-time data on the system performance metrics (through Kafka), the data set stored in the HDFS can be continuously enhanced and, consequently, the models can be retrained through Spark/BigDL within the 5Gr-AIMLP. The model used at the Arbitrator for slice-sharing decision-making can thus be updated accordingly.

SO automatic network service management

This innovation provides the 5Gr-SO with trained ML models that can be used for network service management, e.g., ML-driven scaling in/out of service instances, thus leading to efficient consumption of computational resources. The innovation enables the 5Gr-SO to request the delivery of an ML model for a specific scope and that has been trained using data referring to a specific service type.

SO self-adaptation actions

The innovation enables the 5Gr-SO to dynamically adapt its decision processes to the operational and traffic load conditions. Specifically, the interaction between the 5Gr-SO and the 5Gr-AIMLP allows the latter to identify the appropriate Kafka topics and collect the system performance metrics provided by the 5Gr-VoMS, so as to retrain the ML models and provide an updated version thereof to the 5Gr-SO. Furthermore, the innovation enables the Forecasting block to download an ML model for forecasting the system performance and resource consumption, so that the 5Gr-SO can use forecast values, instead of the current ones, as input to the ML-model provided by the 5Gr-AIMLP for inference.

3.1.6 Innovation 6 (I6): End-to-End Orchestration. Federation and Inter-Domain

3.1.6.1 Architecture

5Growth service providers are able to extend their offering by aggregating the service catalogue and resources from other peering providers through federation and inter-domain support. There are

different types of services that can be handled, namely communication services, network slices, and NFV network services. The communication services and network slices are handled between 5Gr-VSs of two domains. The NFV network services are handled between peering service orchestrators. As for resource federation, it is handled at the service layer (between peering service orchestrators). The diversity of services and technologies leads to new stakeholders and dynamic scenarios where the cooperation requires more rapid interaction. In this context, novel concepts such as Distributed Leger Technology (DLT)-based federation has the potential to bridge the gap and satisfy the emerging requirements.

In this section, we revise and extend the architecture innovation presented in Deliverable 2.1 (Section 4.2.1.6) [1]. On Figure 22, the three architecture building blocks of 5Growth (5Gr-VS, 5Gr-SO, 5Gr-RL) that are involved in federation/multi-domain relations are shown. On the left-side is illustrated a consumer domain, while on the right side, a provider domain. Several options are presented in Deliverable 2.1 (Section 4.2.1.6) [1], and their updates are described below.

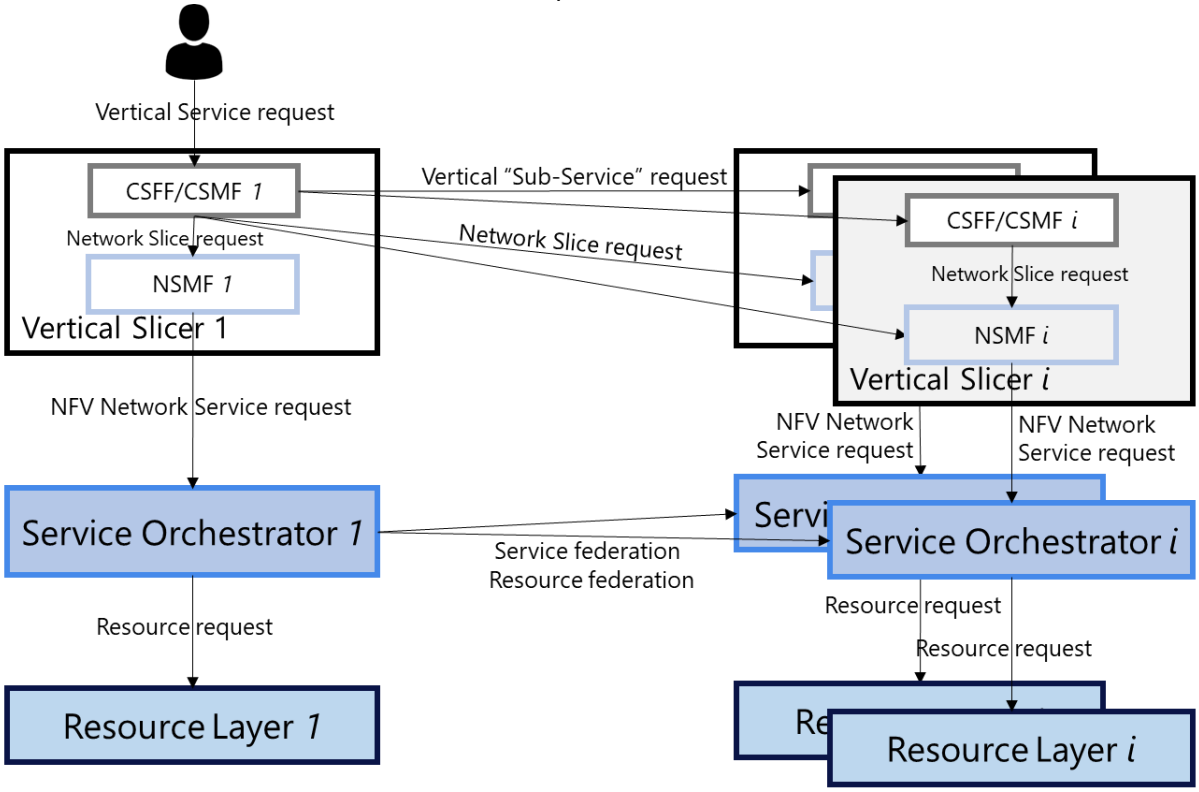


FIGURE 22: FEDERATION/MULTI-DOMAIN ARCHITECTURE

3.1.6.1.1 5Gr-VS updates

To support the multi-domain scenarios at the 5Gr-VS level depicted in Figure 22, namely the multi-domain vertical level split and the network slice level split, the 5Gr-VS information models and functional blocks were updated as follows:

- **Vertical service level split:** In this case, the Vertical Service Blueprint (VSB) information model was updated to allow determining that one of the atomic components of the service is in fact a vertical (sub) service. For these atomic components the correspondent identifier

of the vertical (sub) service and the target administrative domain where to be deployed need to be provided. When the 5Gr-VS receives an instantiation/termination request for the end-to-end vertical service, the vertical (sub)service instantiation/termination request is automatically computed and sent to the target domain using a specific driver registered at the new "Vsmf Interaction handler" module of the 5Gr-VS. Currently, the target domain is defined at the VSB atomic component during the vertical service onboarding. Future work could perform further development to extend the translation capabilities of the 5Gr-VS if required to determine the target domain during the instantiation phase. Moreover, arbitration mechanisms can be extended to support vertical (sub)service sharing among vertical services.

- **Network slice level split:** The outcome of the translation process of the 5Gr-VS was enhanced to allow determining which domain each network slice and network slice subnet shall be requested. The "Nsmf Interaction handler" module was introduced to allow registering the domain-specific drivers, which implement the logic to adapt the network slice lifecycle management request to the specific API provided by the domain. The "Nsmf Interaction handler" processes the incoming requests and forwards the request and uses the mapping between the network slice and the domain (established during the vertical service translation) to forward the request to the appropriate driver.

3.1.6.1.2 NFV network service federation: scaling composite NFV-NS deployments

This subsection connects the SLA management actions in the form of scaling operations for composite NFV-NS deployments with federation aspects considering the NFV network service federation case between 5Gr-SOs described in D2. 1 [1].

The required operations for the scaling process of a composite NFV-NS deployment depends on: (i) how it has been deployed, and (ii) the destination of the scaling request. With respect to the first consideration, a composite NFV-NS can be deployed in either a single administrative domain (AD) or multiple ADs (by using the network service federation (NSF) procedure presented in [31] and sketched in D2.1 [1]). Additionally, such a deployment can be done all-at-once or using a reference to a deployed regular NFV-NS, which may be shared with other composite NFV-NSs. With respect to the second consideration, the scaling operation can be directed to the whole composite NFV-NS deployment or it may be an auto-scaling operation triggered by one of its constituents nested NFV-NSs.

These scaling procedures are connected to the closed-loop operations considered in Innovation 4 (Section 3.1.4) to shape the composite NFV-NS structure based upon network conditions. This allows the 5Gr-SO to handle automatic network service management being compatible with self-adaptation mechanisms. Additionally, in the sharing scenario, if a scaling operation is directed to a shared regular NFV-NS (i.e., non-composite NFV-NS), this operation may imply changes to the associated composite NFV-NSs. All these scenarios have been considered in the 5Growth platform, broadening the scope of the ETSI-NFV IFA028 [34], which, to the best of our knowledge, is one of the first works in the literature considering the scaling operation of composite NFV-NSs deployments. Compared to the 5Growth platform, the ETSI-NFV IFA028 report only considers the service orchestration

perspective of the scaling operation acting at the composite level, disregarding some of the other mentioned scenarios (e.g., those implying auto-scaling operations of nested NFV-NS) and the resource orchestration perspective of the problem to handle the update of inter-nested NFV-NSs connectivity bound to successive scaling operations and possibly involving multiple ADs.

In the 5Gr-SO, the modules handling the deployment of composite NFV-NS are the Service Orchestrator Parent (SOEp) and the Composite Resource Orchestrator Engine (CROOE). Their associated logic and the interface between 5Gr-SOs have been extended to support different scaling situations concerning to composite NFV-NS deployments. Next, the main ideas of the workflows followed at the 5Gr-SO to embrace the diverse situations are presented.

- ***Composite NFV-NS scaling request:*** the SOEp analyses the received scaling request to verify its validity and checks the involved nested NFV-NS with respect to the NSD. Then, it coordinates with the SOEc and other 5Gr-SOs to handle the scaling of the involved nested NFV-NSs deployed in either the consumer or a provider domain[3], respectively. Once, the involved nested NFV-NSs have been scaled, the SOEp coordinates the update of the inter-nested connections between nested NFV-NSs with the CROOE module. The CROOE module, based on the previous set of inter-nested connections, determines the new set. This may entail the creation/termination of new/former connections depending the kind of actions required by the scaling operation (i.e., scale out or scale in) and communicate them with the 5Gr-RL (for interconnections at the consumer domain) or with the CROOE at the other 5Gr-SO (for the interconnections between ADs). These connections form a list of VNF pairs at the different nested NFV-NS sharing the same virtual link. With respect to the deployment procedure, the interface between 5Gr-SOs has been extended to communicate the update of interconnections. This allows specifying the set of interconnections to be created and deleted between VNFs of different nested NFV-NSs as well as the requirements of such interconnections in terms of bandwidth and latency.
- ***Scaling of a shared NFV-NS between composite NFV-NSs*:** when receiving a scaling request concerning a shared NFV-NS, the SOEp realizes about the implied composite NFV-NSs. Then, the SOEp contacts with the SOEc to handle the scaling of the shared NFV-NS as if it was a single NFV-NS. Next, for each involved composite NFV-NS, the SOEp coordinates with the CROOE the update of the interconnections between the shared NFV-NS and the remaining nested NFV-NSs of the involved composite NFV-NSs, both at the consumer and provider domains.
- ***Auto-Scaling of a nested NFV-NS:*** upon detection of an SLA violation in a nested NFV-NS defined at its corresponding NSD (e.g., abnormal CPU value of a VNF within the nested NFV-NS for a time), the SLA manager of the 5Gr-SO triggers an auto-scaling operation.

[3] The consumer domain is the one coordinating the instantiation of the composite NFV-NS. The provider domain is the one satisfying the instantiation request of a nested NFV-NS issued by the consumer domain during the composite NFV-NS deployment involving NSF.

Upon such an event occuring at the consumer domain, the SOEp follows an approach like the previous one to solve the SLA violation. That is, first the SOEp coordinates the scaling of the nested NFV-NS in the provider domain, and then contacts the CROOE to update the inter-nested connections.

If the event happens at the provider domain, the SOEp at the provider domain proceeds to scale the nested NFV-NS and notifies the 5Gr-SO at the consumer domain that a nested NFV-NS is being scaled. The 5Gr-SO at the consumer domain waits until the nested NFV-NS scaling operation finishes to query the new status of the nested NFV-NS, which is required by the CROOE at the consumer domain to update the interconnections between NFV-NS at the different administrative domains.

3.1.6.1.3 Multi-domain support

This subsection describes the Interdomain scenario described in D2.1 [1], which makes use of the changes introduced in the VS to support the 3GPP standards [16][36] on both multi-domain and federation. This scenario envisages the conceptual validation of the necessary mechanisms allowing the service provider to create services by making use of resources available in two different domains (e.g., the case when a vertical requires deploying a single network service into two different operators due to coverage limitations). This is possible as the service provider can use of the VS to create a vertical service which will be associated with a network slice. Following the procedures detailed by 3GPP on network slicing creation and instantiation, this network slice is divided into different sub-slices rolled out in different operators' domains. The realization of such a concept relies on the existence of a communication service level VS that is responsible for the network slice requests, and a network service part which is responsible for the lifecycle operations related with the slice requests. Since the federation scenario covers the sharing of network services at the SO level (i.e., using the 5Gr-SO), and knowing that no network sub slice support existed beforehand, we exploited the decoupling of the CSMF and the NSMF part of the VS to facilitate the instantiation of network sub slices across different domains. The need to make network sub slices requests led to the refactoring of the existent NSMF to support requests to different domains, making use of domain specific drivers. These domain drivers need to interact with an NSMF/NSSMF component that it is present in each domain and which also supports the slice requests.

The Open Source MANO (OSM) or the equivalent SONATA component (available in the Aveiro Test Site of the 5G-VINNI test environment), can both act as the NSMF/NSSMF entity in both domains to test this scenario since they already support network service slicing, inspired by the ETSI NFV-EVE 012 [32] proposal.

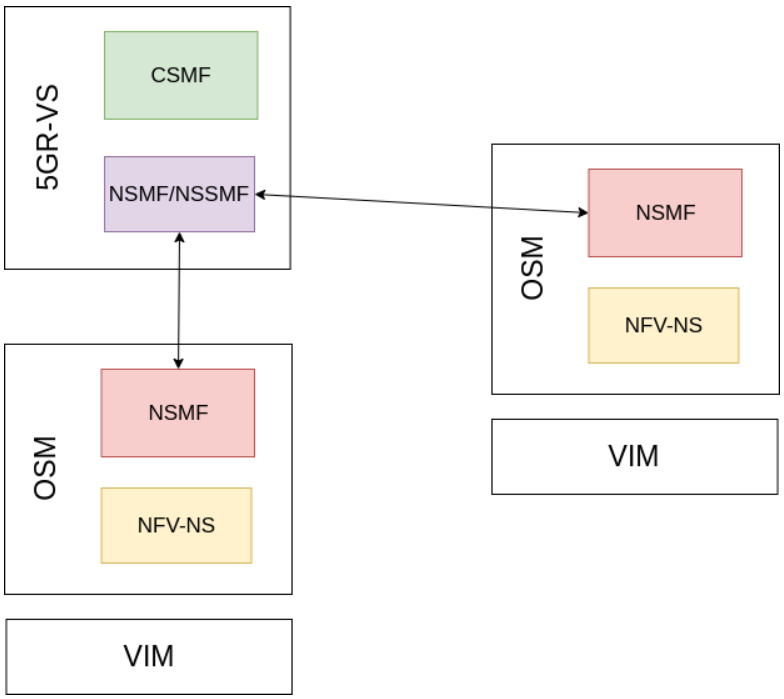


FIGURE 23: INTERDOMAIN CONCEPT GENERAL ARCHITECTURE

Figure 23 depicts the architectural concept for the Network Slice inter-domain solution, where the vertical service generates a slice composed of two sub slices which need to be interconnected at the Network service level through a VPN tunnel (e.g., Wireguard [33]) configured between the 2 VIMs. The purposed process to create the E2E slice across a multi-domain scenario requires that the vertical and the network operators agree about the terms of the slices to be provided. The vertical's employees enter on the vertical's portal information accessing the agreed services and add the information of the network service template identifiers IDs to be used in each domain. The vertical creates a vertical service based on the existent information and triggers the instantiation of such a vertical service. This request indeed triggers the CSMF to instantiate a vertical slice, which will be an E2E slice making use of the NST's provided in the previous step. The 5Gr-VS CSMF divides the E2E Vertical Slice into two sub slices and sends their provisioning requests through the NSMF. The 5Gr-VS NSMF/NSSMF understanding the existence of 2 sub slices provisioning requests will forward those to the respective domain's driver at the same time.

Each domain’s NSMF receives the requests from the NSSMF driver for the creation of a slice following the NST whose identifier was received. This then generates the creation of a network service at the service orchestration level. The SO, upon receiving the request for deploying a network service, seeks for the best VIM to deploy it too and starts the LCM of the network service.

The VIM receives the information of the instantiation requests and allocates the resources to fulfil the request. This entails both day-1 and day-2 configuration of the tunnel which is completed once each network service of the tunnel makes available the information needed for each peer to establish the connection.

When the resource is ready, the LCM within the NFV-NS starts the provisioning of the slice/network services accordingly to the vertical requirements and the computed configurations during the network slice decomposition process of the upper layers in the stack. On-demand configuration is made available if the vertical's services with those primitives exist.

3.1.6.1.4 DLT Federation

The 5Growth platform is able to deploy and orchestrate NFV-NS over multiple administrative domains (ADs) through the federation mechanism. The federation mechanism is used by a consumer domain to deploy network services or allocate of resources over an external provider domain, through a mutual pre-determined trust agreement. Depending on the state of the underlying infrastructure, current orchestration capabilities, specific service geo-footprint (i.e., specific location for service deployment) etc., one AD can decide to deploy part of the service in an external domain. The decision to trigger federation can be executed by the 5Gr-VS or the 5Gr-SO.

Federation interactions can be executed in a centralized, decentralized or distributed manner. A centralized solution means that all involved ADs have to establish mutual agreements and trust a centralized entity located in a neutral location. The centralized entity acts like an intermediary (or a middle-man). A decentralized solution is the simplest one, at the cost of the lowest scalability. An AD establishes peer-to-peer connectivity with each external AD. This implies that each connection is pre-defined with a business agreement and a unique connection towards every agreed AD. On global scale, this impacts the low scalability of the global interworking. A distributed solution is a hybrid approach, where the management is more similar to the centralized solution, however the central entity is distributed in every AD. The joint set-up effort is approximately the same as in the centralized solution with the same benefits and without having a single-point of failure. With the emergence of distributed ledger technologies (DLT) (i.e., blockchain + smart contracts) opens an opportunity for a fast, secure, trusty and efficient setup of highly distributed and scalable federation solution. The feasibility of the DLT solution is presented in Section 4.6.

The idea (shown in Figure 24) is to use a permissioned blockchain where each of the administrative domains runs a single node as part of the permissioned blockchain network. A single generic Federation Smart Contract (SC) is installed to the blockchain to act as a distributed authority. The federation using blockchain provides high degree of security and trust. Each AD running a single node is running the same instance of the Federation SC and each line of code is executed at the same time in all nodes involved in the permissioned blockchain network. The design of the Federation SC is essential for guaranteeing privacy of sensitive information to each AD while overseeing the federation procedures that involve all ADs. Every new AD that joins the blockchain network would need to register to the Federation SC with its unique blockchain address for participation in the federation processes. The registration process includes administrative information of the AD itself and service footprint it provides. This way the AD registration procedure is similar like in the centralized option, executed one-time, scalable and fast.

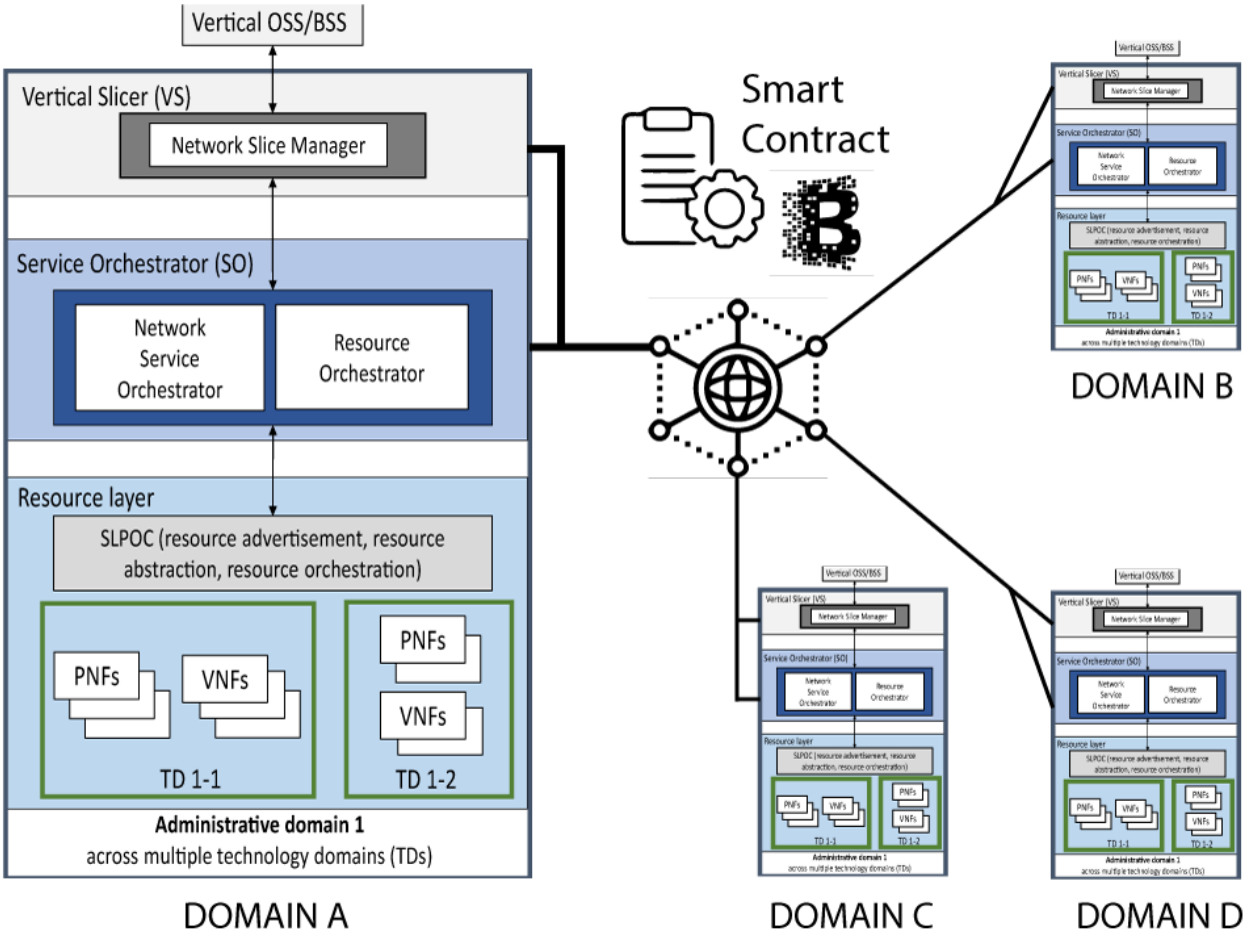


FIGURE 24: FEDERATION USING DLT (BLOCKCHAIN+SMART CONTRACT)

Once all ADs are registered, they can participate in the federation processes as consumers or providers. Figure 25 presents the workflow of the interactions between the ADs with the Federation SC for a single service federation process. When a consumer AD creates an announcement or federation offer, it is sent to the Federation SC which records the offer as a new auction on the blockchain. Then, the Federation SC broadcasts the auction to all registered ADs. Note that the address of the consumer AD is hidden in the broadcast announcement. This is done to protect the AD's privacy, preventing the rest of the ADs to passively collect information. Thus, having in mind that all participating ADs would not have any incentive to fully reveal the federation capabilities, the discovery phase is omitted in the design. Instead, we are using a single-blinded reverse auction [35], where a consumer AD anonymously creates an announcement offer and the rest of the potential provider ADs are bidding for it. Therefore, once the broadcast announcement is received, the potential providers analyse the requirements and place a bid offer to the Federation SC. Each received offer is mapped and recorded by the Federation SC.

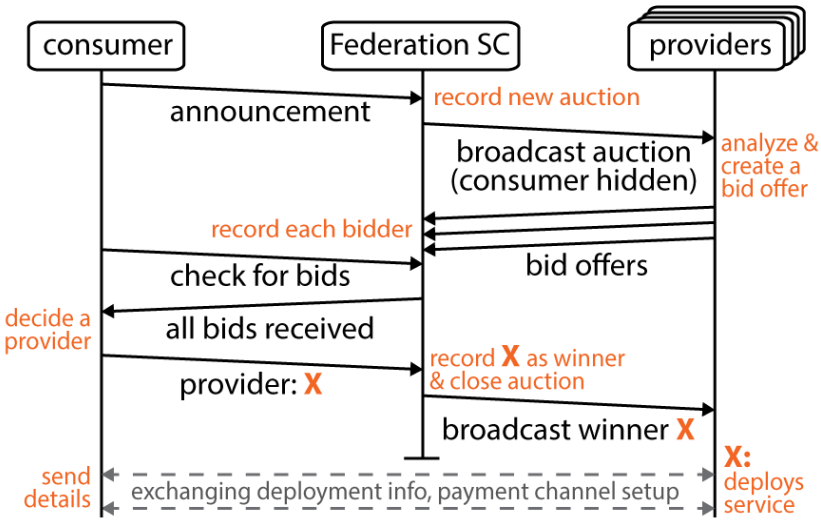


FIGURE 25: SEQUENCE DIAGRAM FOR FEDERATION USING DLT (BLOCKCHAIN+SMART CONTRACT)

Our viewpoint is that the Federation SC should be used as a tool, instead of as an authority in the process. Therefore, the bidding process can only be closed by the consumer AD. In this way the consumer AD has the full control and freedom to apply any selection policies (e.g., prioritize given providers, select the lowest price offer). The consumer AD periodically polls the Federation SC and caches the bidding offers. Once the consumer AD selects a provider AD (e.g., selects the provider X), it closes the auction in the Federation SC. The selected provider X is recorded as a winner by the Federation SC, which then sends a notification to the selected provider X and broadcasts notification to all ADs that the specific auction has finished. At this point, the negotiation and acceptance phases are completed. The consumer AD and the selected provider AD directly communicate and share information. The federated service is deployed and included in the end-to-end service by the consumer AD.

3.1.6.2 Innovation contribution to cover gaps

VS layer federation and VS dynamic service composition

As described before, this innovation established enhanced information models and mechanisms to enable multi-domain scenarios at the 5Gr-VS level addressing the VS layer federation features identified in D2.1 [1]. Moreover, this innovation established the mechanisms for policy-based vertical service to network slice dynamic translations, and settled the basis for future work addressing the dynamic decomposition of vertical service in vertical (sub)services. In addition to the interdomain orchestration, it improves the interaction with verticals by providing extended deployment spectrum of network services on different domains, even if belonging to different operators.

SO self-adaptation actions

This innovation tackles the scaling of composite NFV-NS deployments, even including multi-administrative scenarios. This enhances the automated SLA management capabilities of the 5Gr-SO

also in federated deployment scenarios to react in front of dynamic (and potentially foreseen) changes of network demands and the performance of assigned resources.

SO geo-location dependent federation

Additionally, the federation feature enables for geo-dependent federation. Verticals may request a specific service footprint a given geographical area. If the constituent infrastructure is unable to provide the service coverage, the 5Gr-SO based on the abstracted view from the 5Gr-RL can request a federation with specific requirements. These specific geo-requirements can be requested in any federation realization. For example, the use of DLT would be suitable for more geo-distant federation (e.g., inter-continental, between consumer AD in Europe and provider AD in Asia).

3.1.7 Innovation 7 (I7): End-to-End Orchestration. Next Generation RAN

3.1.7.1 Integration of O-RAN services and interfaces into 5Growth

5Growth follows the architecture of O-RAN to support next-generation RANs. The architecture of O-RAN is presented in Figure 26. Doubtlessly, the most important functional components introduced by O-RAN are the Non-Real-Time (Non-RT) Radio Intelligent Controller (RIC) and the near-RT RIC. While the former is hosted by the Service Management and Orchestration (SMO) framework of the system (e.g., integrated within 5Gr-SO), the latter may be co-located with 3GPP gNB functions, namely, O-RAN-compliant Cloud Unit (O-CU) and/or Distributed Unit (O-DU), or fully decoupled from them as long as latency constraints are respected. The figure also depicts the O-Cloud, an O-RAN compliant cloud platform that uses hardware accelerator add-ons when needed (e.g., to speed up Fast Fourier Transform (FFT) or decoding workflows) and a software stack that is decoupled from the hardware to deploy eNBs/gNBs as virtualized network functions (VNFs) in vRAN scenarios. In the following, we detail the jurisdiction and roles of each functional component defined above.

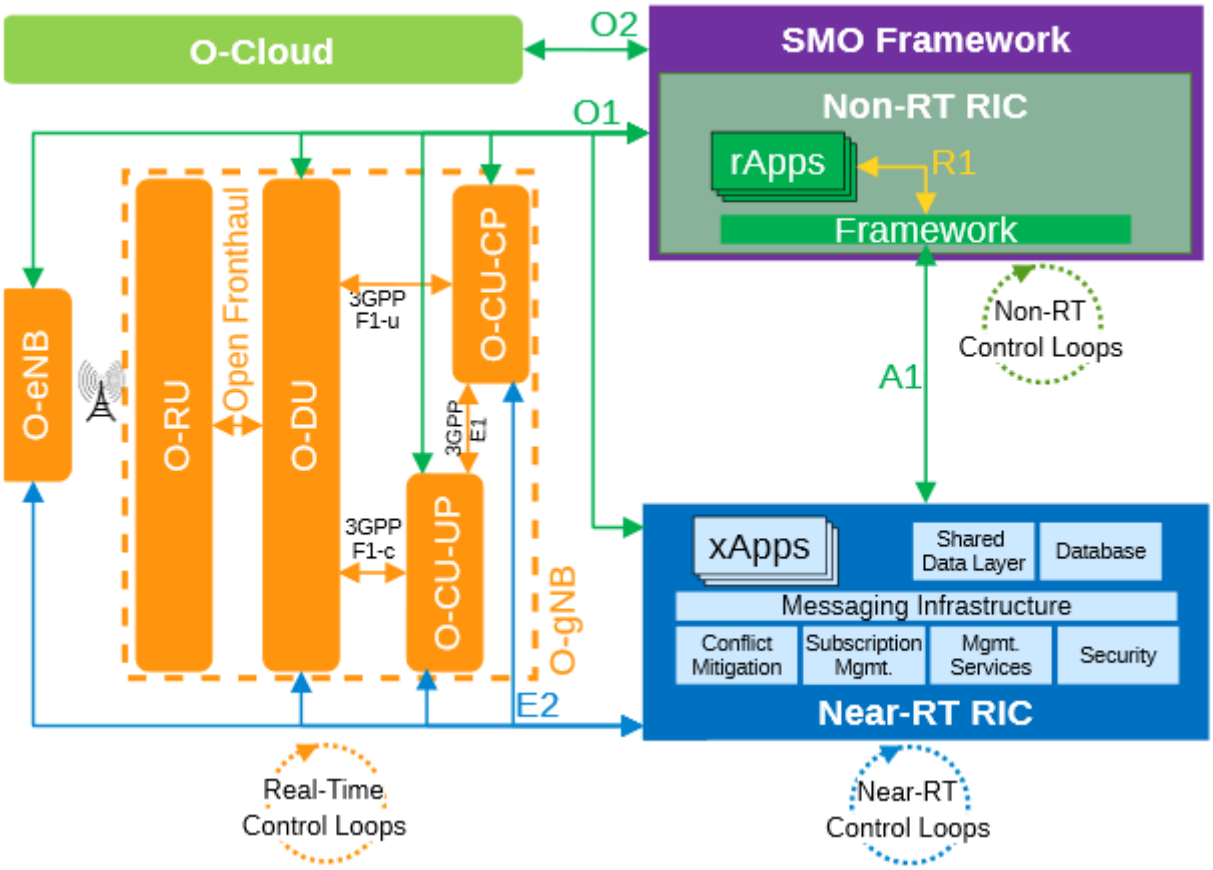


FIGURE 26: O-RAN ARCHITECTURE

- **Service Management and Orchestration (SMO).** The SMO consolidates several orchestration and management services, which may go beyond pure RAN management such as 3GPP (NG-)core management or end-to-end network slice management. In the context of O-RAN, the main responsibilities of SMO are (i) fault, configuration, accounting, performance and security (FCAPS) interface to O-RAN network functions; (ii) large timescale RAN optimization; and (iii) O-Cloud management and orchestration via O2 interface, including resource discovery, scaling, FCAPS, software management, create, read, update, and delete (CRUD) O-Cloud resources.
- **Non-RT RAN Intelligent Controller (Non-RT RIC).** As mentioned earlier, this logical function resides within the SMO and provides the A1 interface to the Near-RT RIC. Its main goal is to support large timescale RAN optimization (seconds or minutes), including policy computation, ML model management (e.g., training), and other radio resource management functions within this timescale. Data management tasks requested by the Non-RT RIC should be converted into the O1/O2 interface; and contextual/enrichment information can be provided to the near-RT RIC via A1 interface.
- **Near-RT RAN intelligent Controller (Near-RT RIC).** Near-RT RIC is a logical function that enables near-real-time optimization & control and data monitoring of O-CU and O-DU nodes in near-real-time timescales (between 10ms and 1s). To this end, Near-RT RIC control is steered by the policies and assisted by models computed/trained by the Non-RT RIC. One of

the main operations assigned to the near-RT RIC is radio resource management (RRM) but near-RT RIC also supports 3rd party applications (so-called xApps).

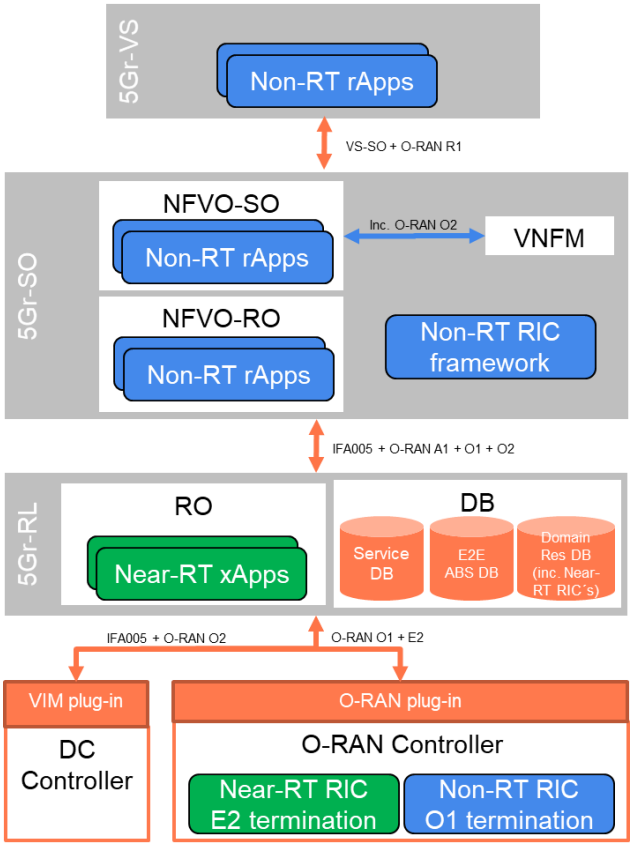


FIGURE 27: INTEGRATION OF O-RAN INTO 5GROWTH ARCHITECTURE

Figure 27 presents the conceptual integration of O-RAN services and interfaces into 5Growth architecture.

Specifically, the Non-RT RIC services are distributed across the 5Gr-VS, 5Gr-SO, and the 5Gr-RL:

- The 5Gr-VS host rApps specifically in charge of business and vertical service related tasks, such as devising RAN coverage area parameters. In order to do so, the rApps will leverage QoS parameters established for the vertical service, slice service type associated with the service and the other specific instantiation constraints, such as the desired coverage area of the service.
- The 5Gr-SO hosts most of the services of the non-RT RIC, including the non-RT RIC framework and the rApps in charge of RAN service and resource related tasks. It also hosts some O2 services to interact with the VNFM in the O-Cloud.
- The 5Gr-RL only hosts the O1 termination endpoint between the Non-RT RIC and the respective managed elements.

Interface R1 is used to let the different rApps interact with the Non-RT RIC framework in the 5Gr-SO. This interface, will be used by the 5Gr-VS to specify the mobile connectivity requirements extracted from the vertical service, following the approach established in Section 3.1.1.1. In particular, to

determine common slice type mobile traffic parameters, such as traffic capacity, coverage area, overall user density etc., and some service slice type specific parameters such as the end-to-end latency for URLLC slices.

All the near-RT RIC services are integrated into the 5Gr-RL, except the E2 termination endpoint, which is offloaded to a dedicated O-RAN controller (different to the RAN controller shown in Section 2.3). In this way, the O-RAN controller handles all the interface endpoints between the RICs and the O-RAN compliant 3GPP radio functions (O-DU, O-CU, O-eNB).

Finally, ETSI NFV IFA005 [8] is to be extended with O2-specific requirements (management of containers and Kubernetes pods). In this way, the same VIM/DC controller introduced in Section 2.3 is used to interact with the O-Cloud.

3.1.7.2 An O-RAN use case: AI-assisted resource orchestration in virtualized RANs (vrAIn)

Dynamic resource allocation in vRAN, illustrated in Figure 28, is an inherently hard problem:

- The computational behaviour of virtualized radio access points (vRAPs), depends on many factors, including the radio channel conditions or users' load demand, which may not be controllable. More specifically, there is a strong dependency with the *context* (such as data bit-rate load and signal-to-noise-ratio (SNR) patterns), the RAP configuration (e.g., bandwidth, MIMO setting) and on the infrastructure pooling computing resources (e.g., speed of the underlying CPUs, presence of task accelerators);
- Upon shortage of computing capacity, e.g., with nodes temporarily overloaded due to orchestration decisions) CPU control decisions and radio control decisions (such as scheduling and modulation and coding scheme (MCS) selection) are coupled; certainly, it is well known that scheduling users with higher MCS incur in higher instantaneous computational load.

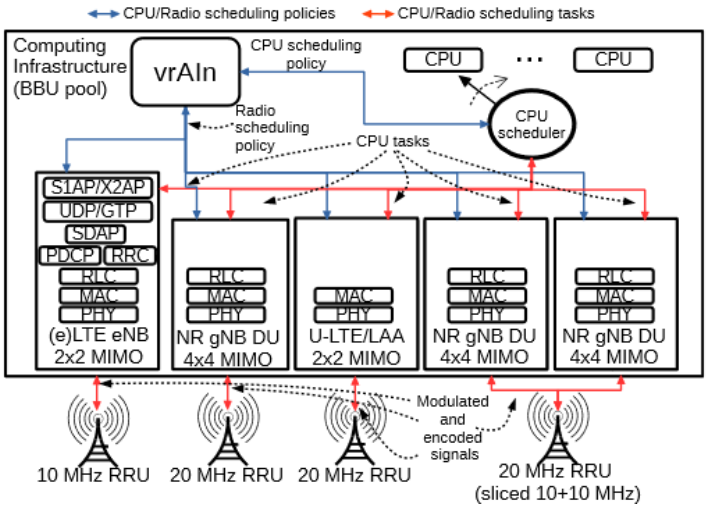


FIGURE 28: VRAIN: A VRAN RESOURCE ORCHESTRATOR

Figure 29 illustrates a model of our system, where we can observe two functional blocks operating at different timescales:

- In the first block, CPU schedulers (which assign tasks to CPUs, e.g., subframes for decoding) and radio schedulers (which assign radio resources, e.g., selecting MCSs) operate at sub-millisecond timescales. vrAIn relies on simple computing and radio to influence their behaviour.
- The second block is vrAIn, our vRAN resource orchestrator, a sequential decision-making entity that configures the above schedulers using, respectively, compute and radio scheduling policies over larger timescales. Our design is compliant with the architecture of O-RAN, which envisions a Non-Real-Time RAN Intelligent Controller operating at second-level granularity. In practice, the operational timescale may be limited by the system constraints, such as the periodicity of feedback information from the users.

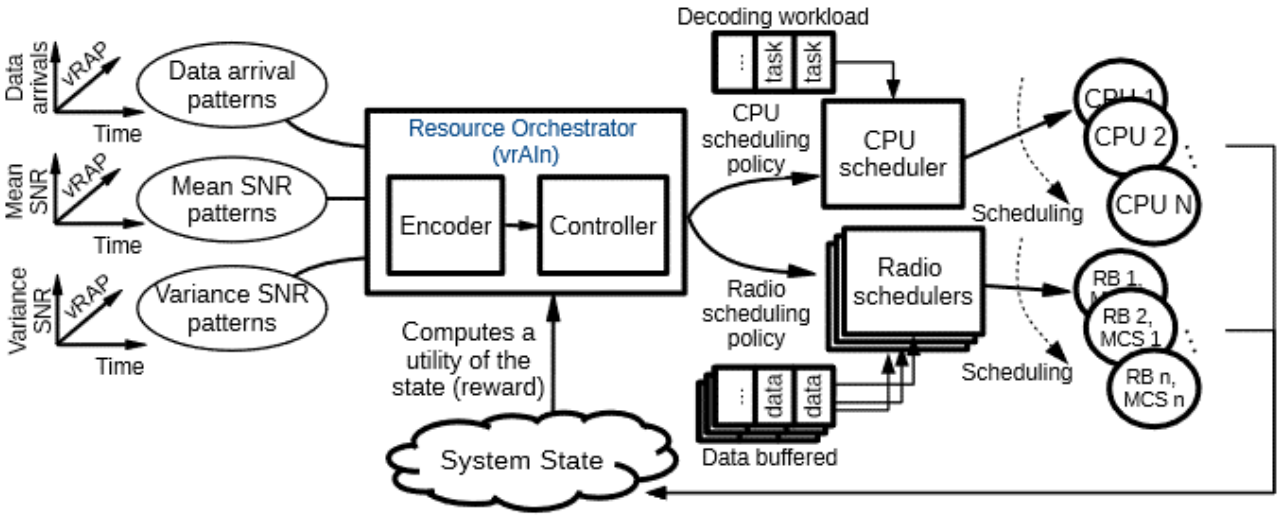


FIGURE 29: VRAIN ARCHITECTURE

vrAIn consists of a feedback control loop where:

- *Contextual* information (SNR and data load patterns) is collected and encoded.
- An orchestrator maps contexts into computing and radio scheduling policies.
- A *reward* signal assesses the decisions taken and fine-tunes the orchestrator accordingly.

This falls naturally into the realm of ***reinforcement learning*** (RL), an area of machine learning applied in human-level control (mastering games such as Go or StarCraft II), health-care or finances. Full-blown RL problems are usually modeled with Markov decision processes and use some model-free learning method (e.g., Q-learning) to estimate an action-value function.

We hence formulate our resource control problem as a contextual bandit (CB) problem, a sequential decision-making problem where, at every time stage $n \in \mathbb{N}$, an agent observes a *context* or *feature* vector drawn from an arbitrary feature space $\mathbf{x}^{(n)} \in \mathcal{X}$, chooses an action $\mathbf{a}^{(n)} \in \mathcal{A}$ and receives a reward signal $r(\mathbf{x}^{(n)}, \mathbf{a}^{(n)})$ as feedback. the sequence of context arrivals $\langle \mathbf{x}^{(n)} \rangle_{n \in \mathbb{N}}$ and the distribution E over context-reward pairs $(\mathbf{x}, r)$ are fixed and unknown *a priori*. Furthermore, we let $\pi(\mathbf{x})$: $\mathcal{X} \to \mathcal{A}$ denote a deterministic that maps contexts into actions, i.e., scheduling policies, and

$$R_\pi := \mathbb{E}_{(\mathbf{x},\mathbf{r})\sim E}[r(\mathbf{x}, \pi(\mathbf{x}))]$$

denote the expected reward of a function π. The goal is to learn an optimal π that maximizes instantaneous reward subject to the system capacity, Π being the space of functions.

Context space. As shown by the experimental study in [38], SNR and traffic load are the contextual features that have most impact on the performance of a vRAP.

Action space. Our action space comprises all pairs of CPU and radio introduced before. Hence, $c_i^{(n)} \in \mathcal{C}$ and $m_i^{(n)} \in \mathcal{M}$ denote, respectively, the *maximum computing time share* (CPU) and the *maximum MCS* (radio) allowed to vRAP i in stage n. We also let $c_0^{(n)}$ denote the amount of CPU resource left unallocated (to save costs). Thus, a resource allocation action on vRAP i consists of a pair $a_i := \{c_i, m_i\}$ and a system action $\mathbf{a} = (a_i)_{\forall i \in \mathcal{P}} \in \mathcal{A} := \{(c_i \in \mathcal{C}, m_i \in \mathcal{M})\}_{\forall i \in \mathcal{P}}$.

Reward function. We design the reward function as follows. Let q_{i,x_i,a_i} be the (random) variable capturing the aggregate across all users of vRAP i given context x_i and action a_i at any given slot. As a quality-of-service (QoS) criterion, we set a target buffer size Q_i for each vRAP. Note that this criterion is closely related to the latency experienced by end-users (low buffer occupancy yields small latency) and throughput (a high throughput keeps buffer occupancy low). Thus, by setting Q_i, an operator can choose the desired QoS, which can be used to, e.g., provide differentiation across network slices. We let $J_i(x_i, a_i) := Prob(q_{i,x_i,a_i} < Q_i)$ be the probability that q_{i,x_i,a_i} is below the target per vRAP i and define reward as:

$$r(\mathbf{x}, \mathbf{a}) := \sum_{i \in \mathcal{P}} J_i\,(x_i, a_i) - M\varepsilon_i - \lambda c_i$$

where ε_i is the *decoding error probability* of vRAP i (which can be measured locally), and M and λ are parameters that determine the weight of decoding errors and the trade-off between resource usage and performance, respectively.

There are several challenges to adopt a CB model, such as continuous and high-dimensional action spaces. These challenges and how vrAIn tackles them are dutifully detailed in [38] As a result, we design vrAIn as a set of interconnected neural networks as shown in Figure 30. The details of each neural network, how they are trained, and how they operate are presented in [38].

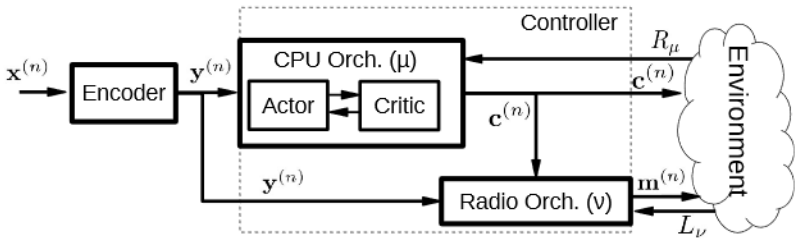


FIGURE 30: RESOURCE ORCHESTRATOR

3.1.7.3 Mapping of vrAIn to O-RAN architecture

The mapping between the above use case (vrAIn) and O-RAN architecture is shown in Figure 31.

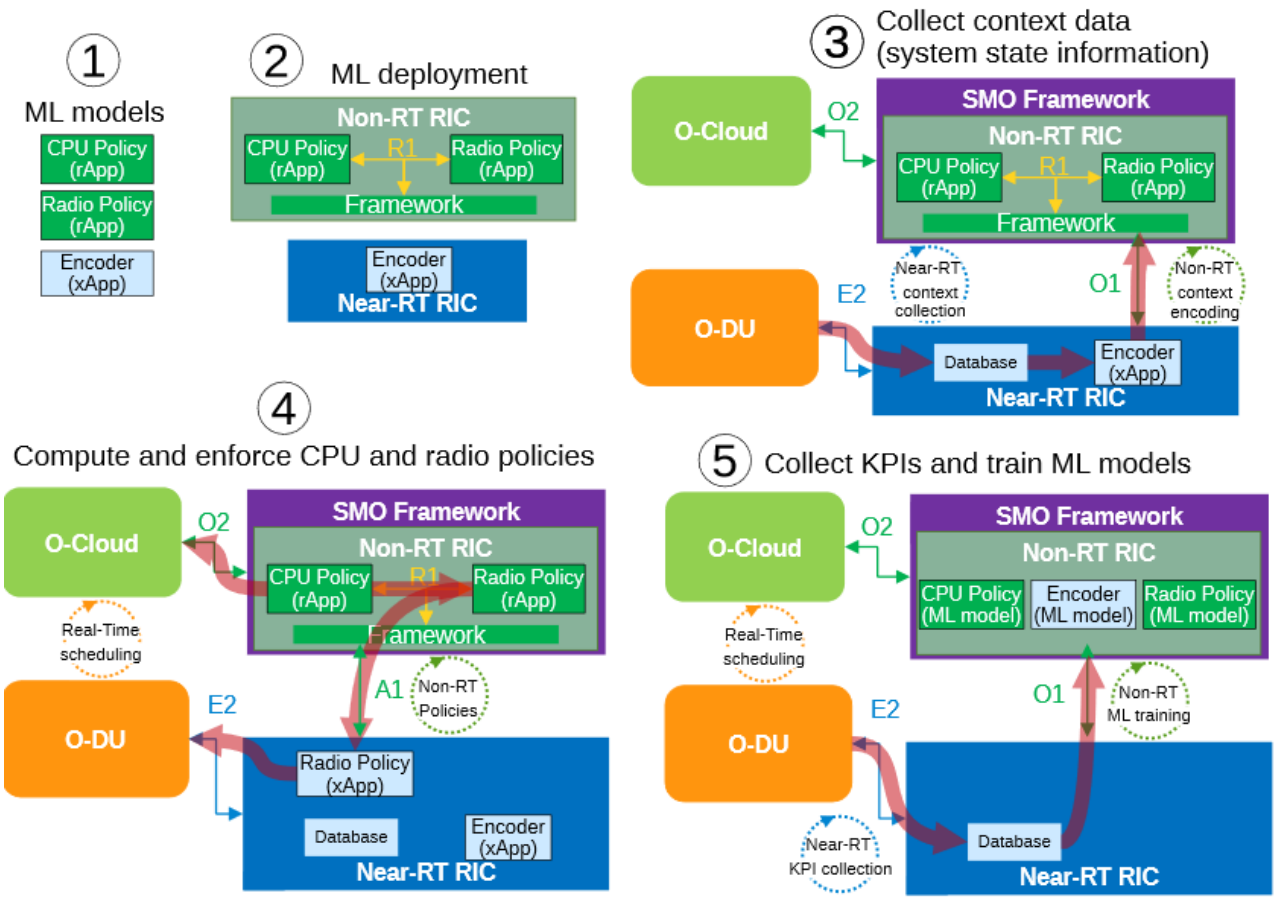


FIGURE 31: MAPPING BETWEEN VRAIN AND O-RAN ARCHITECTURE

We have 3 machine learning models, CPU policy, radio policy and context encoder (see Figure 31 (1) and (2)).

On the one hand, both the CPU actor-critic implementing the CPU policy and the deep classifier implementing the radio policy are hosted by two respective rApps, which communicate through R1 interface, i.e., the Non-RT RIC acts as their inference host and the resulting policies are enforce via O2 (CPU) and A1 (radio) interfaces (see Figure 31 (3)).

On the other hand, the autoencoders implementing our context encoder is deployed into an xApp hosted by the Near-RT RIC, which acts ML inference host (see Figure 31 (4)). During training, which could be done in pre-production (offline), all ML models are trained by the non-RT RIC as shown by Figure 31 (5), with data stored in the near-RT RIC's database

3.1.7.4 Innovation contribution to cover gaps

RL PNF Integration

This innovation integrates O-DU, O-CU, O-RU and O-eNB PNFs as explained by O-RAN specification by incorporating E2, O1 and A1 interfaces.

RAN support

The integration of Open RAN services and interfaces in 5Growth, which is achieved via this innovation opens the door to support the integration of different vendors and novel RAN control and management applications.

3.2 Algorithm Innovations

This section presents the second block of innovations, those related with algorithms that are applied across the 5Growth building blocks entities.

3.2.1 Innovation 8 (I8): Smart Orchestration and Resource Control

Under the umbrella of 5Growth innovation 8, three distinct topics are covered in the following three subsections: (1) Smart orchestration, (2) Resource abstraction and allocation, and (3) Dynamic Profiling Mechanism (DPM). Within each topic, we briefly introduce the core idea of each algorithm innovation and summarize the respective gaps they fill in (as presented in the introduction of Section 3). It is worth mentioning that the focus here is on the innovation from the algorithmic perspective; therefore, each algorithm will be presented in a self-contained manner while their contributions to cover gaps are summarized at the end of each subsection. The corresponding performance evaluation outcomes can be found in Section 4.9.

3.2.1.1 Smart orchestration

In the following, four algorithm innovations for smart orchestration are individually introduced, and we summarize their overall contributions to cover gaps in Section 3.2.1.1.5.

3.2.1.1.1 Genetic Algorithm (GA) based SFC placement

In this section, we briefly introduce the system model and the proposed genetic algorithm for SFC placement, and we encourage the interested reader to refer to our work in [43] for more details.

System Model

In this section, the system model of an SFC, as well as the network topology are analysed. The network physical infrastructure is modelled as an undirected graph $G_p(P, E)$ of Physical Nodes (PNs), where P is the set of PNs and E the set of physical links, assumed to be bidirectional. Each PN hosts a subset of Virtual Machines (VMs), from the set of total VMs V, based on its available resource capacity. It is assumed than the computational resources associated with a PN or a VM are defined in terms of the quadruple: (number of CPU cores, total CPU speed, Memory size, Disk size). Accordingly, the resource capacity of a PN or a VM is denoted by $\{C^*(t)\}_{\{t \in T\}}$, where $t \in T$ ={*number of CPU cores, total CPU speed, Memory size, Disk size*} and * refers to a PN p, $p \in P$, or a VM v, $v \in V$. Let $I(v,p)$ denote a binary variable indicating whether PN $p \in P$ hosts the VM $v \in V$. Each physical link $(p_1, p_2) \in E$, connecting neighbour PNs p_1 and p_2, is characterized by a bandwidth

capacity $b^{(p_1,p_2)}$ and a propagation delay $D_{pr}^{(p_1,p_2)}$, which depends on the transmission medium and the length of the physical link. Let $D_t^{(p_1,p_2)}$ denote the transmission delay, defining the amount of time required to transmit a packet into the physical link $(p_1, p_2) \in E$ and it is inversely proportional to the bandwidth capacity of the physical link $b^{(p_1,p_2)}$.

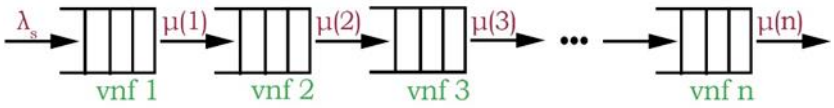


FIGURE 32: SFC MODEL

A service could be represented as a directed graph of VNFs, which is known as SFC, sequentially connected through virtual links (Figure 32). Let S denote the set of all SFCs ready to be placed in the underlying network infrastructure and let F denote the associated VNFs. Each SFC is modeled as a graph $G_s(N_s, E_s)$, where N_s is the set of VNFs of SFC $s \in S$ and E_s is the set of directed edges virtually connecting the VNFs in N_s. Each VNF instance has a certain amount of requested computing resources in T={number of CPU cores, total CPU speed, Memory size, Disk size} notated as $\{R_s^f(t)\}_{t \in T}$. Each virtual link connecting two VNFs $f_1, f_2 \in N_s$ is characterized by a specific requested bandwidth $b_s^{(f_1,f_2)}$. Let $I(f, v)$ denote a binary variable indicating whether VM $v \in V$, hosts the VNF $f \in F$. Let $I(p_1, p_2; f_1, f_2)$ denote a binary variable indicating whether the physical link $(p_1, p_2) \in E$ is part of the physical path between the PNs hosting VNFs f_1 and f_2.The packet arrival rate to each VNF of SFC $s, s \in S$, is modelled by the Poisson process [44] with rate $\lambda_s(f)$, with an average packet size of L, fixed for all VNFs in N_s. According to the aforementioned assumption and in order to estimate the queuing and processing delay of each VNF of a SFCs, each VNF is modeled as an M/M/1 queue, assuming its service time has an exponential distribution. Each VNF is characterized by a service rate $\mu(f)$, in packets per second, proportional to the allocated resource capacity. Rate $\mu(f)$ highly affects the mean processing delay D_{pc}^f (service time) of the VNF $f \in F$, which is measured per packet, since the latter is inversely proportional to the service rate. Let D_q^f denote the time each packet spends in the M/M/1 queue of VNF (queuing delay). Let D_s^{thr} denote the maximum E2E delay (threshold) imposed by the service implemented through SFC $s \in S$; the latter is defined as the sum of all aforementioned delay components (processing, queuing, transmission, propagation). All notations are summarized in Table 8.

TABLE 8: SYSTEM MODEL NOTATIONS

Parameter	Description
G_p	Graph of Physical Infrastructure
P	Set of PNs in G_p
E	Set of bidirectional physical links interconnecting the PNs
V	Set of VMs
T	Set of all resource types
$\{C^*(t)\}_{\{t \in T\}}$	Resource capacity of PN $p \in P$ or VM $v \in V$ for resource type $t \in T$
$I(v, p)$	Binary variable indicating whether PN $p \in P$, hosts the VM $v \in V$
$b^{(p_1,p_2)}$	Bandwidth capacity of physical link between PNs $p_1, p_2 \in P$

$D_{pr}^{(p_1,p_2)}$	Propagation delay of physical link between PNs $p_1, p_2 \in P$
$D_t^{(p_1,p_2)}$	Transmission delay of physical link between PNs $p_1, p_2 \in P$
S	Set of SFCs
F	Set of VNFs modelled as M/M/1 queues
$\mu(f)$	Processing rate of VNF $f \in F$
$\lambda_s(f)$	Packet arrival rate at VNF $f \in N_s$ of SFC $s \in S$
L	Average Packet Length
G_s	Directed Graph of VNFs of SFC $s \in S$
N_s	Set of VNFs of $s \in S$
E_s	Set of of virtual links of SFC $s \in S$
R_f^s	Requested computing resources of VNF $f \in N_s$ of SFC $s \in S$
$b_s^{(f_1,f_2)}$	Requested bandwidth capacity of virtual link between VNFs $f_1, f_2 \in N_s$ of SFC $s \in S$
$I(f,v)$	Binary variable indicating whether VM $v \in V$, hosts the VNF $f \in F$
$I((p_1,p_2);(f_1,f_2))$	Binary variable indicating whether the physical link $(p_1,p_2) \in E$ is part of the physical path between the PNs hosting VNFs f_1 and f_2
D_{pc}^f	Processing delay of VNF $f \in N_s$
D_q^f	Queuing delay of VNF $f \in N_s$
D_{thr}^s	E2E delay threshold imposed to SFC $s \in S$

Problem Formulation

The SFC placement problem is formulated as a mixed-interger linear program (MILP) optimization problem. The goal of the optimization model is to provide a (sub) optimal placement solution towards minimizing the E2E delay of the SFC. Each VNF can be placed on a VM that satisfies the required resource demands. Each virtual link between a pair of VNFs f_1, f_2, denoted as $(f_1, f_2) \in E_s$, can be mapped to a physical path that satisfies its bandwidth requirements $b_s^{(f_1,f_2)}$. A constraint is imposed on the estimated total E2E delay (through the adopted models) of each service to ensure that is upper bounded by the service delay threshold D_s^{thr} provided by the Network Service Descriptor (NSD), to guarantee conformance to the SLA. For each candidate SFC placement solution, if a pair of VNFs is hosted by different PNs and there are multiple paths between the latter, the optimal delay-aware path, in terms of transmission and propagation delay, is selected.

$$min\left(\sum_{f\in N_s}\sum_{v\in V}\left(\left(D_q^f + D_{pc}^f\right)\cdot I(f,v)\right) + \sum_{(f_1,f_2)\in E_s}\sum_{(p_1,p_2)\in E}\left(\left(D_t^{(p_1,p_2)} + D_{pr}^{(p_1,p_2)}\right)\cdot I\left((p_1,p_2);(f_1,f_2)\right)\right)\right) \quad (1)$$

where:

$$D_q^f = \frac{\lambda_s(f)}{\mu(f)\cdot(\mu(f)-\lambda_s(f))} \quad (2)$$

$$D_{pc}^f = \frac{1}{\mu(f)} \quad (3)$$

$$D_t^{(p_1,p_2)} = \frac{L}{b(p_1,p_2)} \quad (4)$$

$$D_{pr}^{(p_1,p_2)} = \beta \cdot l^{(p_1,p_2)} \quad (5)$$

The objective function of our optimization problem for each SFC is the minimization of the service E2E delay and it is formulated and given in Eq. (1). The first part of Eq. (1) expresses the sum of queuing and processing delay for each VNF f, $f \in N_s$,hosted by VM $v \in V$. The second part of the equation expresses the sum of transmission and propagation delays of each physical link $(p_1, p_2) \in E$ that is included in the physical delay-aware path between the PNs p_1, p_2 hosting the VNFs f_1, f_2. The objective function is subject to constraints from Eq. (6) to Eq. (9).

$$\sum_{f \in F} \sum_{v \in V} \left(R_f^s(t) \cdot I(f, v) \right) \leq C^v(t), \forall s \in S, \forall t \in T \quad (6)$$

$$\sum_{(f_1.f_2) \in E_s} \sum_{(p_1,p_2) \in E} \left(b_s^{(f_1.f_2)} \cdot I((p_1, p_2); (f_1.f_2)) \right) \leq b^{(p_1,p_2)}, \forall s \in S \quad (7)$$

$$\sum_{f \in F} \sum_{v \in V} \left(I(f, v) \right) = 1 \quad (8)$$

$$\sum_{f \in N_s} \sum_{v \in V} \left(\left(D_q^f + D_{pc}^f \right) \cdot I(f, v) \right) + \sum_{(f_1,f_2) \in E_s} \sum_{(p_1,p_2) \in E} \left(\left(D_t^{(p_1,p_2)} + D_{pr}^{(p_1,p_2)} \right) \cdot I((p_1, p_2); (f_1, f_2)) \right) \leq D_{\boldsymbol{thr}}^{\boldsymbol{s}} \quad (9)$$

Constraint of Eq.(6) ensures that the resources requested by all VNFs mapped to VM $v \in V$ will not exceed the VM's corresponding resource capacity. Constraint of Eq.(7) ensures that the bandwidth required by all the virtual links $(f_1, f_2) \in E_s$ that include physical link $(p_1, p_2) \in E$, will not exceed that physical link's capacity. Constraint of Eq. (8) ensures that each VNF $f \in F$ will be assigned to one and only one VM and thereby all VNF's requested resources will be satisfied by that VM. Finally, constraint Eq. (9) guarantees that the estimated total E2E delay of each SFC will not exceed the service delay threshold $D_{\boldsymbol{thr}}^{\boldsymbol{s}}$ imposed by the NSD.

Encoding Scheme

In the proposed GA-based optimization model the (sub) optimal placement solution is generated for each SFC of the set S sequentially. A placement solution is represented - and fully identified - by a chromosome (candidate SFC placement solution). A chromosome is represented by a vector consisting of as many genes as the number of VNFs $|N_s|$ of the SFC $s \in S$ that are ready to be placed on the underlying network topology. The position of a gene j ($j \in [1, |N_s|]$) within a chromosome depicts the respective VNF, i, of SFC $s \in S$, represented by this chromosome. The value of the j^{th} gene of a chromosome depicts the VM that hosts the j^{th} VNF of the SFC represented by this chromosome; the range of a gene's value is $[1, |V|]$. At each generation of the GA the set of all chromosomes is referred to as the population of chromosomes and its size is fixed and equal to $population\ size$. The quality of each chromosome is characterized by a fitness value indicating how 'fit' a candidate solution is with respect to the E2E delay of an SFC, the minimization of which is the target of the placement problem. As a result, the fitness function of a is defined in Eq. (1).

GA Operations and Optimization Methods

In the proposed GA-based solution four operations (selection, crossover, mutation, mutagenesis) are applied sequentially in order to generate new chromosomes (offsprings) so as to produce the population of the next generation.

Parent Selection: In each generation, parent chromosomes are selected for crossover in order to create offsprings for the next generations. Each offspring inherits its genes from the two selected

parents. The method considered for parent selection in the GA-based solution is the Roulette Wheel [45].

Crossover: After selecting the two parents, a crossover operation will be applied in order to combine genetic information of the two selected parents to generate two new off-springs. A uniform crossover method is selected.

Mutation: Mutation is the part of the GA which is related to the exploration of the search space. Mutation is applied to randomly tweak genes of a chromosome in order to get a new candidate solution (mutated chromosome). The operation is used to maintain and introduce diversity in the genetic population by altering each gene of a chromosome with a probability mu_rate.

Mutagenesis: After mutation, a mutagenesis operation is applied as a survivor selection policy as described in the paperwork of [46].

GA-based model integrates two further optimization methods, with the first one being a location-aware host selection and the second one a GA's solution filtering, towards further reducing the E2E SFC paths' aggregated delay and GA algorithm's execution time respectively. The location-aware optimization method is proposed, applying limits in terms of network hops (location-aware) between the selected hosts of a SFC. The location-aware optimization method is applied to the set of candidate virtual hosts V. The second optimization method is applied in case of an overloaded network topology. More precisely, when the ratio between the total available and total initial resource capacity is less than or equal to 10%, all overloaded VMs are excluded from the initial set of candidate hosts in order to avoid unnecessary or even non-operational time overheads.

Genetic Algorithm

In this section, a GA-based meta-heuristic approach tailored to the problem formulation is proposed. A meta-heuristic-based approach is selected instead of a heuristic-based one, since the latter has typically a narrower search scope and it is more prone to stagnation in local optima [47].

For each SFC, an initial population of candidate VNF placement solutions satisfying the imposed constraints from Eq. (6) to Eq. (9), is randomly generated (Alg.1 in Figure 33, ln.5). For each chromosome of the population a delay-aware path is selected, and the location-aware method is applied aiming at further minimizing the E2E delay of a SFC (Alg.1 in Figure 33, ln.10-12). The fitness value of Eq. (1) is calculated for each chromosome of the population. The two fittest chromosomes, which are known as *elites*, are instantly included in the new population for the next generation in order to ensure that they will not be altered by any of the GA operations (Alg.1 in Figure 33, ln.15). Additionally, for each chromosome a parent selection operation is applied in order to choose two parents for crossover. For each pair of parents two new offsprings are produced and all constraints are checked to ensure they are satisfied. If not, the crossover operation is repeated for the same pair of parents until producing offsprings satisfying the constraints. To avoid infinite loops, a termination criterion is applied after a predefined number of loops l and the pair of parents are added to the new population as offsprings (Alg.1 in Figure 33, ln.16-18 & Alg.2 in Figure 33, ln.1-9).

After the crossover operation, the set of all new offsprings are mutated and a mutagenesis operation (Alg. 4 in Figure 33) is applied to the two worst-fit chromosomes. To enhance potentially slow convergence under overloaded conditions, a filtering method is applied that excludes all overloaded VMs from the initial set. The aforementioned process is repeated for the next generation until convergence to the (sub) optimal delay-aware SFC placement solution. The loop of generations is terminated when either the maximum number of generations has been reached or the best-scored chromosome has not changed for a certain number of generations, denoted by initial threshold (Early Stopping); the latter is typically chosen to be a small fraction of the total number of generations. To avoid potential deceleration due to the initial threshold value selection, an adaptive threshold thr is introduced, whose value is decreased by a step size st in each generation that does not change the fitness value for the best-scored (Alg.5 in Figure 33, ln.1-3). If thr decreased to zero then the current best-scored chromosome is selected as final solution (Alg.5 in Figure 33, ln.4-6). If GA produces a new best scored chromosome, then the threshold is reinitialized and the process is repeated (Alg.5 in Figure 33, ln.8). Alg.1-5 summarize the aforementioned steps. The corresponding performance evaluation outcomes can be found in Section 4.9.1.1.

Integration with the 5Growth End-to-End Platform

The GA-based SFC placement mechanism is responsible for the delay-aware VNF placement and the selection of a delay-aware path (in terms of transmission and propagation delays) between each pair of VNFs of a SFC. The 5Gr-SO receives a unified view of the underlying network topology (with a suitable abstraction) along with the available resources from the 5Gr-RL that are necessary for the implementation of the GA-based resource allocation mechanism. The (sub)optimal SFC placement solution, provided by the GA-based mechanism, is exploited by the 5Gr-SO towards managing the placement of the respective VNFs at the 5Gr-RL and requesting the appropriate resource allocation from the 5Gr-RL.

Algorithm 1: Delay-aware SFC placement

Input: Virtual, Physical Network topology; SFC; population size; no. of generations, elites, k,l, pc
Output: Delay-aware SFC placement
create network topology graph $G_p(P, E)$
create SFC topology graph $G_s(N_s, E_s)$
randomly initialize population P with size equal to *population size*;
initial threshold = $\frac{no.of generations}{k}$
thr = *initial threshold*
st = *initial threshold* $\times pc$
while *generation* $\leq$ *no.of generations* **do**
  foreach *chromosome chr in population P* **do**
    find delay-aware path (10) with location-aware host selection enabled
    calculate fitness value (11)
  end
  sort population P in an increasing order
  select elites chromosomes and add to new population P_{new}
  foreach *chromosome chr in population P* **do**
    */*Parent Selection Operation*/*
    select two parents using ***Roulette Wheel method***
    */*Crossover Operation*/*
    $P_{new} \leftarrow$ ***Crossover(l, parents)***[*Alg.* 2]
  end
  foreach *chromosome chr in population P* **do**
    */*Mutation Operation*/*
    $P_{new} \leftarrow$ ***Mutation(l, chr)***[*Alg.* 3]
  end
  Mutagenesis(P_{new}, ***l***) *[Alg. 4]*
  Early Stopping(P_{new}, ***thr, st, l***) *[Alg. 5]*
end

Algorithm 2: Crossover

while *no. crossover attempts* $< l$ **do**
  apply *uniform* crossover to selected parents
  check constraints (6) - (9) for the two new offsprings
  if *offspring solutions are valid* **then**
    add offsprings to P_{new}
    break
  no. crossover attempts+=1
if *no. crossover attempts* $= l$ **then**
  add parents to P_{new}
return P_{new}

Algorithm 3: Mutation

while *no. mutation attempts* $< l$ **do**
  apply mutation to *chr*
  check constraints (6) - (9) for the mutated *chr*
  if *mutated chr is valid* **then**
    add mutated *chr* to P_{new}
    break
  no. mutation attempts+=1
if *no. mutation attempts* $= l$ **then**
  add *chr* to P_{new}
return P_{new}

Algorithm 4: Mutagenesis

sort population P_{new} in increasing order
while *no. mutagenesis attempts* $< l$ **do**
  apply mutagenesis to two worst-fit chromosomes
  check constraints (6) - (9)
  if *two worst-fit chromosomes are valid* **then**
    update the two worst-fit chromosomes
    add to P_{new}
    break
  no. mutagenesis attempts+=1

Algorithm 5: Early Stopping

sort population P_{new} in increasing order
if *best score chromosome same with previous generation* **then**
  decrease thr by st
  if *thr is equal to zero* **then**
    select fitness value of best-score chromosome as final solution
    break
else
  initialize thr to *initial threshold*

FIGURE 33: PROPOSED GENETIC ALGORITHMS

3.2.1.1.2 VNF Autoscaling

The continuously changing network dynamics of 5G and beyond networks, call for continuous monitoring of the network traffic load and scaling accordingly the network services. To this end, the current section proposes a Deep Neural Network (DNN)-based scheme, and specifically a Multi-Layer

Perceptron (MLP)-based one, for proactively scaling services, according to the respective network requirements and real-time traffic load.

Control Model

The proposed MLP-based scheme predicts the IL, in terms of the number of required VNF instances per VNF type, in order to accommodate the traffic load changes ahead of time and satisfy the respective network service demands.

The current AI-driven mechanism investigates how to map traffic load statistics X_s of a service $s \in S$ served via a specific base station, to instantiation levels Y_s of the VNF, in a supervised manner. Let traffic load statistics $x_t \in X_s$, denote a vector consisting of traffic load measurements, as well as user equipment (UE)-related information, in a specific period of time t. In addition, let $y_t \in Y_s$, denote the IL required in order to proactively accommodate traffic load demands of the next time period $t+1$. The target of the MLP model is to learn a function $f: X_s \rightarrow Y_s$ so that $f(x)$ accurately predicts the corresponding value of y.

The proposed MLP-based scheme, which is based on the work of [48], considers a set of data features (X_s), related to the traffic load of each service and how this load evolves over time. The selected features are listed below:

- 5G New Radio gNodeB Base Station ID (denoted gNB ID)
- Average number of associated, active UEs
- Traffic load in packets and bytes (for each gNB ID) for five time periods ($[t-2, t+2]$)
- Change in traffic load in packets and bytes (received by cell) for the five aforementioned time periods.

For each service, an MLP model is deployed in order to identify relevant patterns, while mapping features X_s to autoscaling decisions Y_s. The output class of each MLP model refers to the service IL required to meet traffic demands until the next scaling decision (after 5 time periods). In case the host does not have enough capacity to meet the predicted service IL for the VM, then the maximum supported IL is selected.

The corresponding performance evaluation outcomes can be found in Section 4.9.1.2.

Integration with the 5Growth End-to-End Platform

For the proactive service scaling, through the proposed MLP-based model solution (described in the "Control model" paragraph), the interaction between the 5Gr-AIMLP and the 5Gr-SO is required. More specifically, the MLP-based model (trained in the 5Gr-AIMLP, using the already gathered dataset from the 5Gr-VoMS), is requested to the 5Gr-AIMLP and installed in the SLA manager of the 5Gr-SO. Through the SLA manager, the MLP model is fed with all necessary data features towards proactive decision making (IL of the NFV-NS). In case a different IL is predicted by the MLP model the SLA manager triggers a scaling operation (scale in/out).

3.2.1.1.3 AI-Driven scaling of C-V2N Services

We consider a C-V2N (cellular vehicle-to-network) environment where cars exchange information with the cloud via Points of Presence (PoPs) that are located throughout a city. Whenever a car enters into the neighborhood served by a PoP, its workload needs to be supported by the computational resources of that PoP. These resources come in discrete chunks expressed as the number N of CPUs (central processing unit) that have been activated at a certain moment in time. It is the job of the 5Gr-RL to determine that number of CPUs, as cars travel through the city, and hence, as the workload that is presented to that PoP fluctuates over time.

To derive the number of CPUs over time $N(t)$, we propose the use of two AI-Driven solutions, namely: (1) Reinforcement Learning (RL), and (2) Forecasting + Scaling solutions.

In particular, the used RL solution corresponds to a Q-learning agent that says whether to scale up $N(t+1) = N(t) + 1$ or down $N(t+1) = N(t) - 1$ the number of CPUs, using the system status at time t. Using a Q-table, the RL agent uses information about the current load of each CPU $W(t) \in \{0,1\}^{N(t)}$, to derive its scale-up, or scale-down decisions.

The proposed Forecasting + scaling solution uses a LSTM Neural Network to anticipate what would be the future demand $W(t+1)$, and scale the number of CPUs to satisfy the forecasted workload.

Both solutions have been compared against a Proportional Integral (PI) controller that keeps the most loaded CPU $\max_i W_i(t)$ around a threshold ρ_{tgt}, therefore, it estimates the required increase in the number of CPUs $\delta(t)$ to derive the scaling decision as $N(t+1) = N(t) + \delta(t)$. On top, the CPU load and reward of the mentioned solutions were compared against a benchmark approach with a constant number of CPUs over time $N(t) = \Delta, \forall \Delta \in \mathbb{N}^+$. The corresponding performance evaluation outcomes can be found in Section 4.9.1.3.

3.2.1.1.4 Slice sharing algorithm at the Arbitrator

The algorithm presented here enables service instances simultaneously running in the 5Gr-infrastructure to share sub-slices whenever possible. So doing, the creation of unnecessary sub-slice instances is avoided, thus increasing the efficiency in the usage of computing resources.

In a nutshell, the algorithm takes as input the latency classes to consider, with a latency class being defined as a range of target service latency values. The number of possible classes and the width of each class depend on the number of running services on the infrastructure. The latency classification to be used is an input parameter to the algorithm and is provided by an ML model run at the Arbitrator. Upon the arrival of a deployment request for a new service instance, the algorithm identifies the corresponding latency class and checks whether any of the already deployed slices or subslices can be resused. A (sub-)slice can be shared if all of the following conditions are met: (i) there must be no service isolation constraints, the (sub-)slice has to belong to the same latency class as that of the newly requested service, the computing resources necessary for the (sub-)slice to handle both service traffic loads must not exceed the maximum value that the (sub-)slice can get assigned.

A detailed description of the algorithm follows.

The algorithm is executed at the VS Arbitrator every time a new service has to be instantiated. A generic sub-slice $\hat{v}$ included in the VNF Forwarding Graph (VNFFG) of the new service, s_1, can be shared with another service, s_2, including $\hat{v}$ as well, if both s_1 and s_2 fall in the same latency class for $\hat{v}$. The latency class of a VNF depends on its target processing latency, as well as the service target latency.

Figure 34 presents the algorithm pseudocode, while Table 9 reports the notation used therein.

TABLE 9: NOTATION USED IN THE PSEUDOCODE OF THE SLICE SHARING ALGORITHM

Symbol	Description
$\bar{\mu}$	Maximum computing capability of a VM
θ_v	Complexity factor of VNF v indicating the amount of vCPU required by v to process a traffic unit
$\mathcal{V}_s$	Set of VNFs composing service *s*
D_s	Target latency for service *s*
$d_{l^*,r}$	Network latency between the target user(s) and network layer l^*
λ_r	Expected traffic load to be processed by service request instance *r*

The algorithm takes as input: (i) the newly requested service instance, along with the set $\mathcal{V}_s$ of VNFs composing the service, the expected service traffic load λ_r, and the service target latency D_s; (ii) the maximum computing capability $\bar{\mu}$ associated with a single VM; (iii) the per-VNF *complexity factor* θ_v indicating the amount of virtual CPU (vCPU) required to process a traffic unit by the VNF v; (iv) the network latency $d_{l^*,r}$ between the end user and the layer l^* of the fat-tree topology at which the new service should run. Then, the algorithm initializes two sets: $\mathcal{V}_r$ to set $\mathcal{V}_s$ which contains the VNFs composing the new service (Line 2), and $\mathcal{O}$ to the empty set (Line 4). The latter will include the algorithm output; each item in $\mathcal{O}$ is a tuple composed of three elements: the service instance identifier, a VNF v, and a VM capability μ_{b^*} needed to run VNF v.

For each subslice v in $\mathcal{V}_r$ (Line 5), the minimum processing latency (Line 7), the target latency (line 9), and the latency class (Line 11) are computed. Then, the algorithm looks for VMs already instantiated that can be reused for request r. In particular, all the VNFs $\hat{v}$ composing service instances ρ among the running services $\hat{\mathcal{R}}$ are analysed (Line 14 and 16). Line 18 reports the control needed to determine whether the current VNF $\hat{v}$ can be shared with the newly service: if $\hat{v}$ is one of the VNFs requested in r, and both their placement and latency class match, then $\hat{v}$ can potentially be shared. If a VNF is shared with other services, the capability of the corresponding VM, say b^*, may need to be adjusted, till a maximum capability $\bar{\mu}$. If the current capability of VM b^* is lower than $\bar{\mu}$ (Line 20), in Line 21 the algorithm computes the new capacity μ_{b^*}. Line 22 adds the tuple "service instance id – VNF – capability", i.e., (ρ, $\hat{v}$, μ_{b^*}), to the set $\mathcal{O}$. Finally, Line 23 removes $\hat{v}$ from the set $\mathcal{V}_r$, whereas Line 26 checks if $\mathcal{V}_r$ is empty. If so, all the VNFs composing the new service have been processed, and, consequently, the algorithm ends returning set $\mathcal{O}$. If a VNF cannot share an existing VM, then a new VM is created and the necessary computing resource is allocated.

Algorithm 1 Sub-slice reuse

Require: Service instance request $r = \langle \mathcal{V}_s, D_s, \lambda_r, \ell^* \rangle$, $\bar{\mu}$, θ_v, $d_{l^*,r}$

▷ Given request r, initialize $\mathcal{V}_r$ to the set of VNFs composing the corresponding service
$\mathcal{V}_r \leftarrow \mathcal{V}_s$
▷ Define $\mathcal{O}$ (what the algorithm will return) as empty set
$\mathcal{O} \leftarrow \emptyset$
for all $v \in \mathcal{V}_r$ **do**
    ▷ Compute the min processing delay for VNF v of request r
    $M_{s,v} = \frac{1}{\bar{\mu} - \theta_v \lambda_r}$
    ▷ Compute its target latency
    $D_r^v(\ell^*) \leftarrow \frac{M_{s,v}}{\sum_{u \in \mathcal{V}_s^r} M_{s,u}} (D_s - d_{\ell^*,r})$
    ▷ Determine its *latency class*
    $j_v^* \leftarrow \lfloor log_{(1+\epsilon)} D_r^v(\ell^*) \rfloor$
end for
▷ For each service instance ρ of service σ among the running instances $\hat{\mathcal{R}}$
for all $\rho \in \hat{\mathcal{R}}$ **do**
    ▷ For each VNF $\hat{v}$ composing the current service instance ρ
    for all $\hat{v} \in \mathcal{V}_\sigma$ **do**
        ▷ Check if $\hat{v} \in \mathcal{V}_r$, their service level and *latency class*
        if $\hat{v} \in \mathcal{V}_r$ **and** $\ell^\rho = \ell^*$ **and** $j_v^* = j_v^\rho$ **then**
            ▷ Adjust capability of the VM hosting the VNF $\hat{v}$ (b^* denotes the VM and $\hat{r}$ is the generic service instance running and using b^*)
            if $\mu_{b^*} \leftarrow \theta_v[\Lambda(b^*) + \lambda(r)] + \frac{1}{\min_{\hat{r}} D_{\hat{v}}^{\hat{r}}(\ell^*)} \leq \bar{\mu}$ **then**
                $\mu_{b^*} \leftarrow \theta_v[\Lambda(b^*) + \lambda(r)] + \frac{1}{\min_{\hat{r}} D_{\hat{v}}^{\hat{r}}(\ell^*)}$
                $\mathcal{O} \leftarrow \mathcal{O} \cup (\rho, \hat{v}, \mu_{b^*})$
                Remove v from $\mathcal{V}_r$
            end if
            ▷ Check if all the VNFs composing the new service are instantiated
            if $\mathcal{V}_r = \{\}$ **then**
                break
            end if
        end if
    end for
end for
▷ If at least one VNF v has not been instantiated yet
if $\mathcal{V}_r \neq \{\}$ **then**
    for all $v \in \mathcal{V}_r$ **do**
        ▷ Create a VM with the following capability
        $\mu_{b^*} \leftarrow \theta_v \lambda(r) + \frac{1}{D_v^r(\ell^*)}$
        $\mathcal{O} \leftarrow \mathcal{O} \cup (r, v, \mu_{b^*})$
    end for
end if
return $\mathcal{O}$

FIGURE 34: PSEUDOCODE ON THE SLICE SHARING ALGORITHM

In summary, the slice sharing algorithm at the 5Gr-VS Arbitrator enables for an efficient usage of computational resources, by allowing for slice (or sub-slice) sharing whenever possible. Conditions under which slice sharing can be applied depend on the service isolation constraints and required target KPI values. The slice sharing algorithm, in particular, tackles latency requirements, and an efficient allocation of computing resources that can meet such requirements by keeping the processing time sufficiently low, while limiting service deployment costs. The corresponding performance evaluation outcomes can be found in Section 4.9.1.4.

3.2.1.1.5 Innovation contribution to cover gaps

SO automatic network service management, SO self-adaptation actions

The aforementioned algorithms in the above sections, i.e., GA-based SFC placement, VNF scaling, and scaling for a C-V2N service respectively, can fill these two specific gaps.

Specifically, the GA-based SFC placement algorithm in Section 3.2.1.1.1 provides a solution towards placing VNFs of the respective SFC, considering both resource-related requirements (CPU, memory, disk storage, etc.) as well as service-related KPIs (service E2E delay). As a result, the output of this algorithm could potentially be retrieved by the 5Gr-SO in order to place and allocate the appropriate resources requested by each VNF of the SFC and perform the mapping between the virtual and physical links for each pair of VNFs.

Moreover, the AI-driven automated scaling solution in Section 3.2.1.1.2 exploits the MLP DNN algorithm and it can monitor and analyse (1) network and service traffic-related measurements and (2) service requirements. The final output of the MLP-based automated scaling is the service IL, in terms of the number of required VNF instances per VNF type, in order to accommodate the traffic load changes ahead of time and satisfy the respective network service demands. The service IL, produced by the automated scaling solution, could be retrieved by the 5Gr-SO towards automated service management ensuring no SLA violation and self-adaptation by allocating the appropriate resources indicated by the IL.

Further, the AI-driven scaling of C-V2N services of Section 3.2.1.1.3, monitors the available CPU resources and decides to switch on or off CPU resources, if the CPUs get overloaded or underutilized respectively, such that no cost is incurred for not delivering an adequate service or for having to many CPU resources activated. The number of CPUs needed for the C-V2N service as calculated by the scaling algorithm, could be used by the 5Gr-SO to enable automated service management and to apply self-adaptation.

VS dynamic service composition, VS arbitration at runtime

Regarding the slice sharing algorithm provided in Section 3.2.1.1.4, it covers these two gaps. Such algorithm already addressed the VS dynamic service composition feature we identified in D2.1 [1], showing how an algorithm was developed to determine possible network slice instance sharing among different vertical service instances. In this sense, given that vertical services can now be decomposed into several vertical (sub)services, future work may follow a similar approach to enable smart vertical (sub)service sharing mechanisms. Moreover, future work may also leverage the algorithms established at the 5Gr-SO level for the resource allocations to provide enhanced mechanisms for vertical service to vertical (sub)service and network slice translation. In particular, to better determine at runtime the dimensions of the different vertical (sub) services and network slices and the domains to which these different service components should be requested.

3.2.1.2 Resource abstraction and allocation

In the following, two algorithm innovations for resource abstraction and allocation are introduced, and we summarize their contributions to cover gaps in Section 3.2.1.2.3.

3.2.1.2.1 5Gr-SO - 5Gr-RL: Resource Abstraction and Allocation Algorithms

This section addresses two proposed implementations supporting the interactions between both the 5Gr-SO and 5Gr-RL entities when deploying a new received network service (NFV-NS). Both implementations differ on: (i) the resource (i.e., computing and networking) abstraction made at the 5Gr-RL and eventually exposed to the 5Gr-SO; and (ii) the resource selection to accommodate the NFV-NS fulfilling the demanded requirements (e.g., network bandwidth or end-to-end latency). The devised solutions are referred to as Infrastructure Abstraction (InA) and Connectivity Service Abstraction (CSA).

As mentioned, both CSA and InA consider diverse approaches for producing the NFVI abstraction embracing the Logical Links (LLs) creation between pairs of NfviPops. Another difference is the resource granularity level used by either CSA or InA solutions to perform the resources selection for accommodating the targeted NSes.

- In the InA approach, the exposed LLs by the 5Gr-RL are associated to explicitly feasible WAN paths between each NfviPop pair. This allows the 5Gr-SO selecting the NfviPops (hosting the NFV-NS's VNFs) and the corresponding LLs.
- In the CSA model, the set of LLs are bound to a pre-determined set of connectivity service types/classes (e.g., Gold, Silver and Bronze) offered by the 5Gr-RL. Each connectivity service type defines their own features (e.g., minimum guaranteed bandwidth, maximum tolerated delay). Once the 5Gr-SO selects the NfviPops and LLs, the 5Gr-RL needs to conduct specific WAN path computations (i.e., expansion function) to fulfil the service type parameters associated to every LL.

Prior to describe the CSA and InA characteristics and operations, it is worth detailing the overall abstraction and expansions functionalities adopted in the 5Growth stack, especially between the 5Gr-SO and 5Gr-RL. Figure 35 depicts the adopted NFVI view handled by both the 5Gr-SO and 5Gr-RL entities. An NFVI embraces two resource types: i) computing resources (i.e., CPU, RAM, and Storage) within an NfviPoP; and ii) transport network (WAN) resources for the NfviPops inter-connections. For the computing resources, the 5Gr-RL (interacting with the VIM controllers) retrieves (in an aggregated way) the total and available amount of CPU, RAM, and Storage at every NfviPop. The same computing resource view is kept by the 5Gr-SO.

For the transport network resources, the view at both 5Gr-SO and 5Gr-RL elements differs. The 5Gr-RL interacts with the WIM controller/s to create the WAN network topology (nodes and link connectivity). For each link, the 5Gr-RL is aware of a set of attributes characterizing the network link such as the available bandwidth (in b/s), the associated delay (in ms), the cost, etc. On the other hand, the abstracted WAN view at the 5Gr-SO is formed by the LLs as shown in Figure 35. Indeed, the LLs

constitutes a virtual representation of the (potential) physical connectivity between a pair of NfviPops. Every LL has its own attributes in terms of bandwidth, delay, cost, etc.

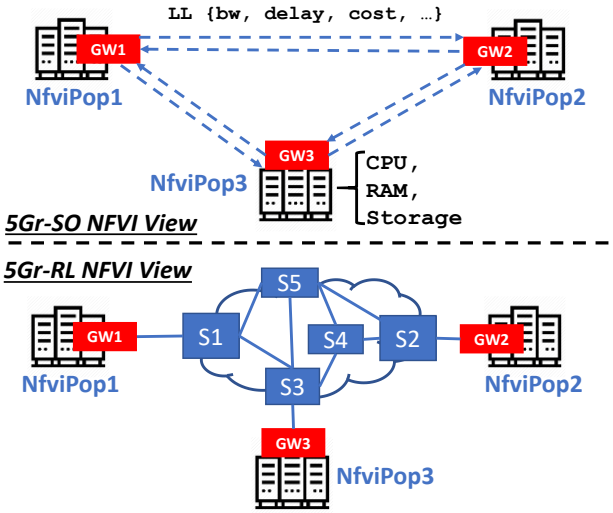


FIGURE 35: 5GROWTH STACK RESOURCE VIEW

The adopted 5Gr-RL operational mode (either InA or CSA) determines how the LLs are derived and then exposed to 5Gr-SO:

- In the InA mode, every LL is always associated to a feasible computed WAN path realized at the 5Gr-RL. Thus, the LL attributes inherit the features of the underlying WAN path supporting such an LL (e.g., unused bandwidth on the most congested path link, total accumulated delay over each traversed WAN path link). By doing so, note that once the 5Gr-SO selects a specific LL for the NS, it implicitly determines the WAN nodes and links to accommodate the traffic flow between the respective NfviPops.
- In the CSA mode, the LLs are not bound to a pre-computed WAN path. Indeed, the 5Gr-RL defines a set of service types/classes which define the supported characteristics of the candidate NfviPop pair connectivity. In other words, no explicit abstraction computation is triggered to derive the exposed LLs to the 5Gr-SO. It is assumed that three connectivity service types (i.e., Gold, Silver and Bronze) are offered by the 5Gr-RL. Each type/class provides its own parameters, guaranteed bandwidth, and maximum delay. Conversely to the InA, once the 5Gr-SO selects a LL for the NS, the 5Gr-RL is then required to conduct an explicit WAN path computation (i.e., expansion function) to find the nodes and links supporting the selected LL attributes, i.e., the associated service type/class parameters.

InA Workflow and Algorithm

The workflow implementing the InA operational mode is depicted in Figure 36 and starts once the 5Gr-SO receives a new NSReq. An NSReq specifies an NS descriptor detailing the set of VNFs (with the required amount of CPU, RAM, and Storage) as well as the VLs to be deployed among the VNFs. Typically, the VLs specifies network-oriented requirements such as bandwidth (in b/s) and the maximum latency/delay (in ms).

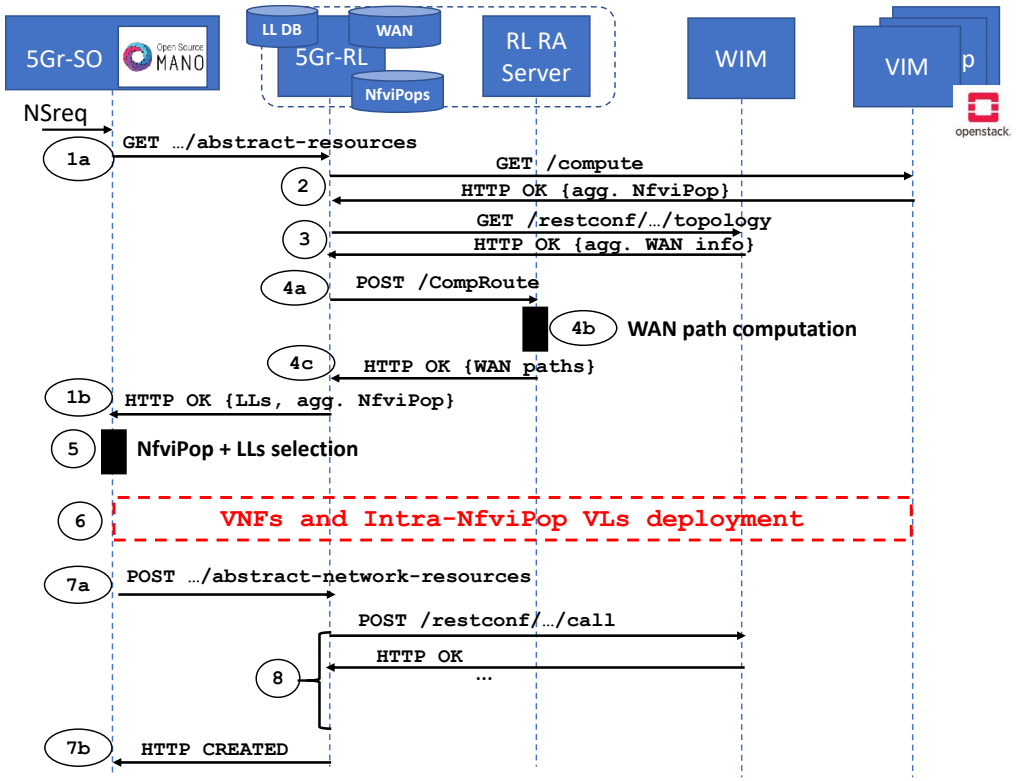


FIGURE 36: WORKFLOW IN THE 5GROWTH STACK FOR RA INA

To serve an incoming NSReq, the 5Gr-SO first asks (step 1a) to the 5Gr-RL the NFVI abstraction view (i.e., NfviPops and LLs resources). This is done via a RESTful API. For the NfviPop resources, the 5Gr-RL communicates (step 2) with every NfviPop controller (VIM) to retrieve the aggregated amount of CPU, RAM, and Storage. Similarly, the 5Gr-RL requests to the WIM information about transport WAN, i.e., nodes and links' attributes (step 3). The received WAN information is then used as input parameter to query the RL RA server (step 4a) for computing (up to K) WAN paths to interconnect every possible NfviPop pair within the NFVI. The WAN path computations are done by the devised InA algorithm at the RA server (step 4b). This algorithm basically seeks for the K WAN paths which are sorted according to different prioritized criteria: i) the largest end-to-end available bandwidth, ii) shortest end-to-end delay, and iii) lowest aggregated cost between each pair of NfviPops. The resulting (K) WAN paths for each NfviPop pair derives the set of LLs (and their attributes) being exposed to the 5Gr-SO (step 1b). After receiving the abstracted NFVI view, the 5Gr-SO triggers (step 5) a placement algorithm (out of the scope of this work) to choose the NfviPops (hosting the VNFs) along with the LLs enabling the NfviPop connectivity. The latter are used to accommodate the VLs between VNFs at different NfviPops. In step 6, the 5Gr-SO (using OSM MANO) coordinates the VNF deployment at the selected NfviPops through communicating with the VIMs. Next, the selected LLs are notified (step 7a) to the 5Gr-RL. In this request, it is specified the amount of bandwidth to be allocated on each LL. Note that this requested bandwidth is associated to the requirements of the VLs. For each selected LL, the 5Gr-RL picks its previously computed WAN path (done in step 4b). The workflow is completed in step 8, where the flows on the involved WAN paths (i.e., nodes and links) are programmed by the corresponding WIM controller.

The offered advantage bound to the InA operational mode is that the placement mechanism at the 5Gr-SO operates with a set of LLs whose underlying WAN paths are feasible. This may favour the 5Gr-SO LL selection mechanism to better fulfil the NS requirements. However, this is attained at the expenses of asking the 5Gr-RL computing all the WAN paths to derive the LLs; depending on the underlying WAN size this may lead to experience scalability issues in terms of high computation burden, increased NFV-NS deployment times, etc.

CSA Workflow and Algorithm

The basic idea behind the CSA operational mode is that the exposed LLs are bound to specific and pre-defined connectivity service types (e.g., Gold, Silver and Bronze) that can be offered to be deployed over the WAN. Thus, in the CSA a dedicated abstraction computation is not needed to be conducted (at the 5Gr-RL) to infer the LLs. However, once the 5Gr-SO selects the LLs (and thus a specific connectivity service type, e.g., Silver) to accommodate an NFV-NSreq, the 5Gr-RL seeks for the WAN paths that fulfil those LL requirements (e.g., those associated to the Silver connectivity service type). This path computation made by the 5Gr-RL is defined as the expansion mechanism/function.

Figure 37 depicts the workflow for the CSA operational mode. In step 1a, the 5Gr-SO requests the NFVI abstraction to the 5Gr-RL. NfviPop resources are directly asked to the VIMs' NfviPops as shown in step 2. As for the LLs, the 5Gr-RL advertises its pre-defined LL database mapping the LLs with the supported connectivity service types. Both NfviPop and LLs are exposed to the 5Gr-SO in step 1b, which then executes the placement mechanism (step 3). The NSReq's VNFs are deployed at the selected NfviPops (step 4). After that, the 5Gr-SO commands the 5Gr-RL allocating the WAN resources to fulfil the selected LLs' needs. For each of these LLs, the steps 6 – 8 are sequentially repeated. First, in step 6 the 5Gr-RL updates its WAN view. After this, the RA CSA algorithm is queried to find the WAN path fulfilling the LL requirements (steps 7a). This request contains the LL's ingress and egress NfviPops, the bandwidth and latency requirements, along with the retrieved WAN view. The RA CSA algorithm, executed in step 7b, outputs the first path satisfying the requirements between the ingress and egress NfviPops. Finally, such an output (WAN path) is processed by the 5Gr-RL to eventually conduct the flow programmability over the path (step 8).

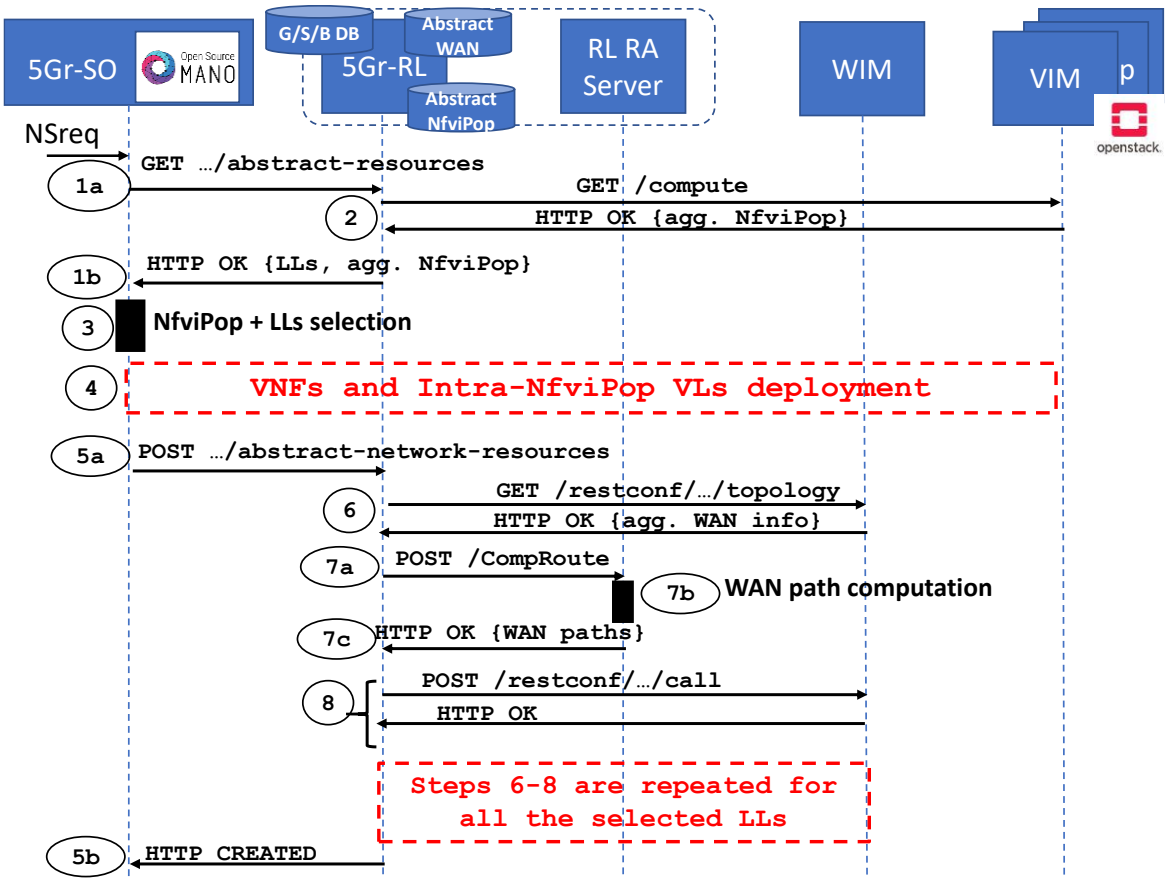


FIGURE 37: WORKFLOW IN THE 5GROWTH STACK FOR RA CSA

The CSA operational mode does reduce the computational burden at the 5Gr-RL to infer the LLs (i.e., the abstraction mechanism). This, however, has two implications: i) the selected LLs made by the 5Gr-SO when serving a NFV-NSReq may not be eventually allocated by the 5Gr-RL. This happens if the expansion function done by the CSA algorithm is unable to find a feasible path satisfying the LL requirements; ii) the CSA expansion algorithm is executed for each individual LL chosen by the 5Gr-SO. To do so, the 5Gr-RL must interact with the WIM (to update the WAN view) before performing each path computation. The latter does increase notably the 5Gr-RL - WIM interactions. The corresponding performance evaluation outcomes can be found in Section 4.9.2.1.

3.2.1.2.2 Performance isolation approach on network slicing to retain bandwidth and delay guarantee

Performance isolation in SDN-based network slicing is a relatively recent research topic. Specifically, several related works are using SDN in conjunction with different technologies to implement traffic metering and traffic prioritization that are the key features beside performance isolation. Among these prior arts, we can classify them according to their applied approaches, i.e., OpenFlow or P4, and briefly review in the following.

The works in [49] and [50] use OpenFlow meters and transmission queues to provide bandwidth guarantees; however, these solutions are not integrated with the concepts of network slicing. In particular, [49] utilizes a multi-table pipeline, where the first table is used to direct the packets

towards the meters for distinguishing the traffic in three QoS categories, and the second table is used to direct the three traffic categories toward different transmission queues. Conversely, [50] proposes and implements an extension of OpenFlow meters together with a meter rate evaluator mechanism to support a hierarchy of meters. Exploiting the defined hierarchy, the threshold of a meter is dynamically determined at the device level, based on actual traffic measured by higher priority meters. Additionally, the authors of [3] propose a framework to deploy a set of network slices using OpenFlow protocol. Specifically, an ONOS SDN controller is used to create and deploy network slices providing both connectivity and performance isolation in terms of guaranteed bandwidth. The deployed solution not only utilizes the Virtual Private LAN Service (VPLS) application to create connectivity-isolated multipoint-to-multipoint network slices, but also introduces the QoSlicing application to enforce bandwidth threshold for each slice using OpenFlow meters.

On the other hand, P4 specification provides a standard method for bandwidth management using packet classification and metering in [51], in which packet classification uses "two rate Three Color Marker" (trTCM) scheme standardized by IETF in RFC 2698 and stateful meter object is used to record statistics of a traffic flow. This information can be utilized to perform different actions, i.e., mark or drop packets, according to burstiness and bit rate criteria. Compared with OpenFlow, it has been argued in [52] that P4 meters are more flexible in terms of band type (i.e., associated operation); however, P4 meter are statically defined by the P4 program with programmable trTCM parameters. Nevertheless, meters are only exposed to P4 programmers as an extern function that can be used to call metering built-in functionality, which may not be available for all P4 targets, as in the case of NetFPGA-SUME, or may have proprietary or limited behaviour as in the case of Netronome SmartNICs that use a single rate/burst pair of limits and thus can report only two states for a packet.

Therefore, in this section, we introduce the novel concept of virtual queues in the P4 pipeline, as shown in Figure 39. In short, the virtual queue is a mechanism used to model the sojourn latency of the queue, as if the packet arrives at the real queue, serving by a link with a capacity lower than the actual capacity of the link. Such modelled sojourn latency is also termed as "virtual delay", in comparison with the real delay spent in the P4 pipeline, which can be used to enforce traffic policing or to even drive the Active Queue Management (AQM). To be specific, we excerpt the "rate limit" action apply by the P4 program in Figure 39, in which we highlight some key aspects:

- The *slice_ts* register stores the global timestamp of the previous packet of the slice (lines 7-13).
- The *slice_delay* register holds the slice's virtual queue size in terms of delay (line 19) and is updated (line 30) once the virtual queue grows.
- Depending on the time that has elapsed and the inter-packet arrival time, the waiting time of current packet is determined (lines 20-24).
- If the size of the virtual queue, augmented by the transmission delay (T_DELAY), exceeds the drop burst limit (C_DELAY), then the packet is dropped. Otherwise, the virtual queue size is incremented by the transmission delay (lines 25-29).

- If the TCP version at the flow level supports Explicit Congestion Notification (ECN) functionality, then the packet is ECN-marked when the virtual queue delay exceeds the marking burst limit (M_DELAY) in order to indicate the congestion (lines 32-34).

It is noted that such virtual queue is implemented using commonly supported P4 constructs (i.e., registers), so that it can be straightforwardly ported to different P4 targets with a similar performance (in terms of processing delay) to the P4 meters available on the Netronome SmartNIC. The corresponding performance evaluation outcomes can be found in Section 4.9.2.2.

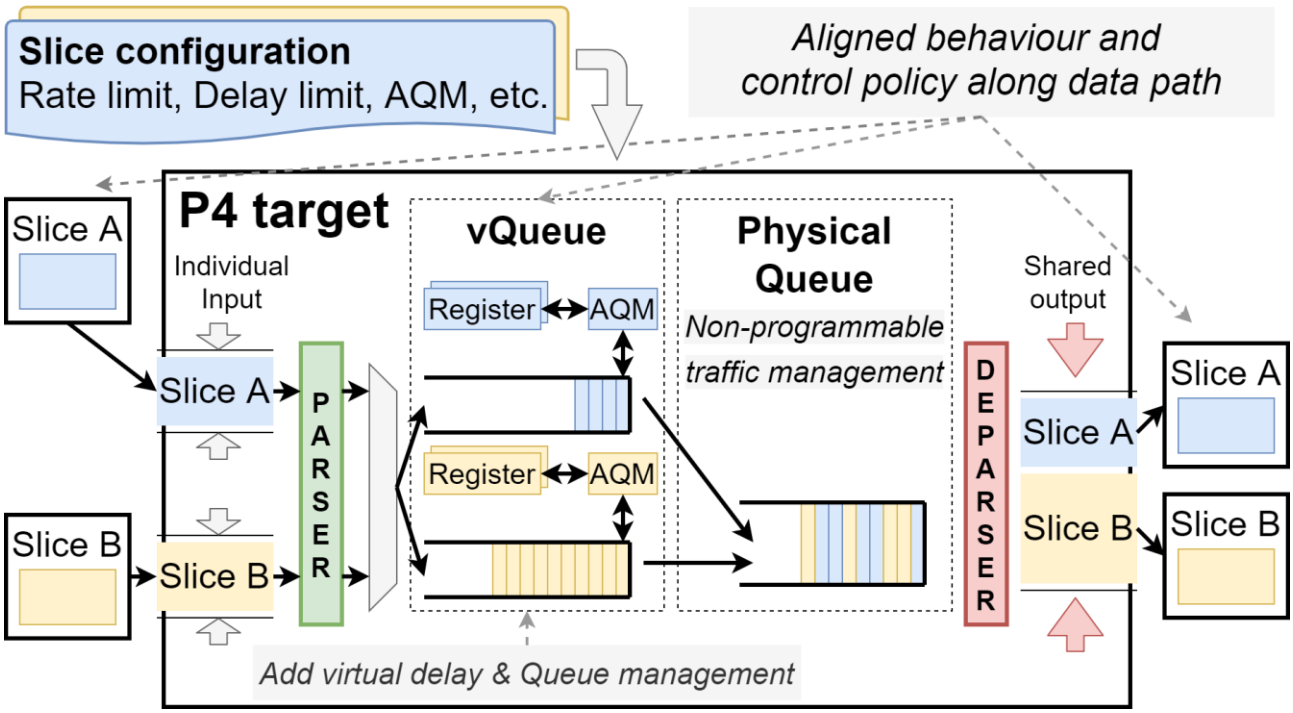


FIGURE 38: VIRTUAL QUEUE IMPLEMENTATION OVER P4 PIPELINE

```
action rate_limit(bit<64> T_DELAY, bit<64> C_DELAY, bit<64>
    M_DELAY){
  bit<64> delay=0;                                  //Delay
    reported
  meta.ts=hdr.intrinsic_metadata.current_global_timestamp;
  bit<64> c_ts = meta.ts;
  bit<64> p_ts;
  bit<64> delta = 0;
  @atomic {                                         //Update
    timestamps
    slice_ts.READ_REG(p_ts,meta.slice_id);
    slice_ts.WRITE_REG(meta.slice_id, c_ts);
  }
  if ((p_ts==0) || (p_ts>c_ts)) {                   //For reordered
     packets
      p_ts = c_ts;
  }
  delta = c_ts-p_ts;
  if (delta >= 3294967296) {                        //Wrap up
    timestamps
      delta=delta-3294967296;
  }
  @atomic {                                         //Update delay
      slice_delay.READ_REG(delay,meta.slice_id);
    if (delta > delay) {
        delay = 0;
    } else {
      delay = delay - delta;
    }
    if (delay + T_DELAY > C_DELAY) {
      meta.DropFlag = 1;                             //Drop Packet
    } else {
      delay = delay + T_DELAY;
    }
    slice_delay.WRITE_REG(meta.slice_id,delay);
  }
  if ((meta.flag == 0) && (hdr.ipv4.ecn != 0) && (delay >
    M_DELAY)) {
      hdr.ipv4.ecn = 3;                             //Mark Packet
  }
}
```

FIGURE 39: RATE LIMIT P4 ACTION

3.2.1.2.3 Innovation contribution to cover gaps

SO automatic network service management

The algorithms described in Section 3.2.1.2.1, i.e., tackling diverse 5Gr-SO and 5Gr-RL interactions for resource abstraction and allocation, aim at mainly filling this gap. More specifically, two different operational modes (CSA and InA) are proposed at the 5Gr-RL. These two operation modes target diverse mechanisms differing on not only the adopted abstraction algorithm but also on the resulting accuracy-level of the abstracted resources exposed to the 5Gr-SO. As a result, depending on the applied operational model, the decisions made by the 5Gr-SO when automatically serving and deploying network services are considerably different, e.g., requiring the 5Gr-RL to trigger further computations (i.e., resource expansion selection) leveraging its more granular view of the underlying resources.

As for the second algorithm mentioned in Section 3.2.1.2.2, i.e., performance isolation approach on network slicing to retain bandwidth and delay guarantee, it complements this gap via providing different underlying infrastructure resources, i.e., virtual queue and virtual delay, to serve the needs for slice-specific delay and bandwidth guarantee without extra tuning. To be more specific, it relies on both resource orchestrator and WIM plugin within 5Gr-RL to add the delay-related parameters within the slice-specific QoS policy for slice orchestration.

3.2.1.3 Dynamic Profiling Mechanism (DPM)

3.2.1.3.1 Introduction to the DPM

The implemented algorithmic framework, namely Dynamic Profiling Module (DPM) provides a detailed overview of a novel mechanism, which monitors and combines (processes) big volumes of diverse network- and user/device- oriented data and extracts profiles for UE and Things, based on past behaviour in terms of device type, mobility patterns, service consumption, etc. In order to achieve the latter, the DPM processes contextual information that is aggregated from diverse network entities and segments as well as logging modules. Ultimately, the aforementioned extracted knowledge is exploited in order to forecast network and service requirements, as well as UE-related behaviour (e.g., device mobility, service consumption). As a result, this extracted knowledge could be exploited by respective proactive network management and resource allocation schemes, generating considerable gains for the network.

3.2.1.3.2 Data Modelling

This section provides the details of the data model designed and used by DPM. Different data categories - from now on denoted as *data entities* - are identified, namely *Device*, *Service* and *Network*. All entities are associated to a specific UE, which is characterised by a unique identifier, - i.e., the International Mobile Subscriber Identity (*IMSI*)-. Each one of the three data entity types is correlated with the other two, since each single UE is using a *Device* that executes a *Service* through a *Network Radio technology*. A visual representation of the data model is given in Figure 40. The *Device* entity refers to a specific type of equipment that a user is using and comprises of data features that are related to geolocation (latitude, longitude), user velocity, battery status of the device, radio signal-related metrics (Reference Signal Received Power/Quality - RSRP/RSRQ and transmission power), as well as user-cell association information. The *Service* entity is related to the type(s) of active session(s) of the user. It is described by the service name, the transfer protocol used for the Transport Layer (TCP/UDP), as well as a set of service-related metrics, namely Uplink (UL) and Downlink (DL) packet size and inter-packet transmission interval. The *Network* entity refers to the type of Radio Access Technology (RAT) serving the specific user-cell association. The identified data features of *Network* entity are related to traffic load and end-to-end delay for both UL and DL data flows (transmitted and received data in packets/bytes, UL/DL end-to-end delay). Besides, cell-related information is included, namely the transmission power of the base station, the type of the cell, the allocated bandwidth, as well as the number of the associated UEs.

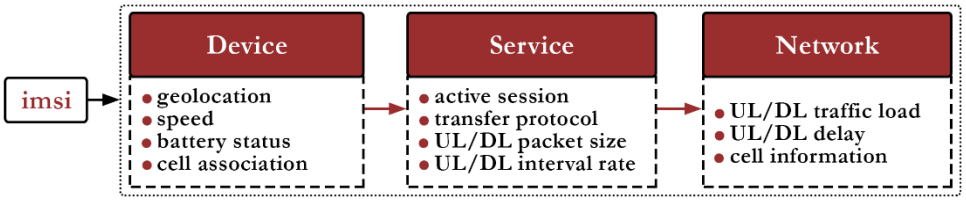


FIGURE 40: DATA MODEL OF DPM

3.2.1.3.3 Overview and Methodology of the DPM framework

DPM's step-by-step methodology (Figure 41), as well as the list of modules, which comprise the overall framework, and their respective functionality are presented in detail in the following section.

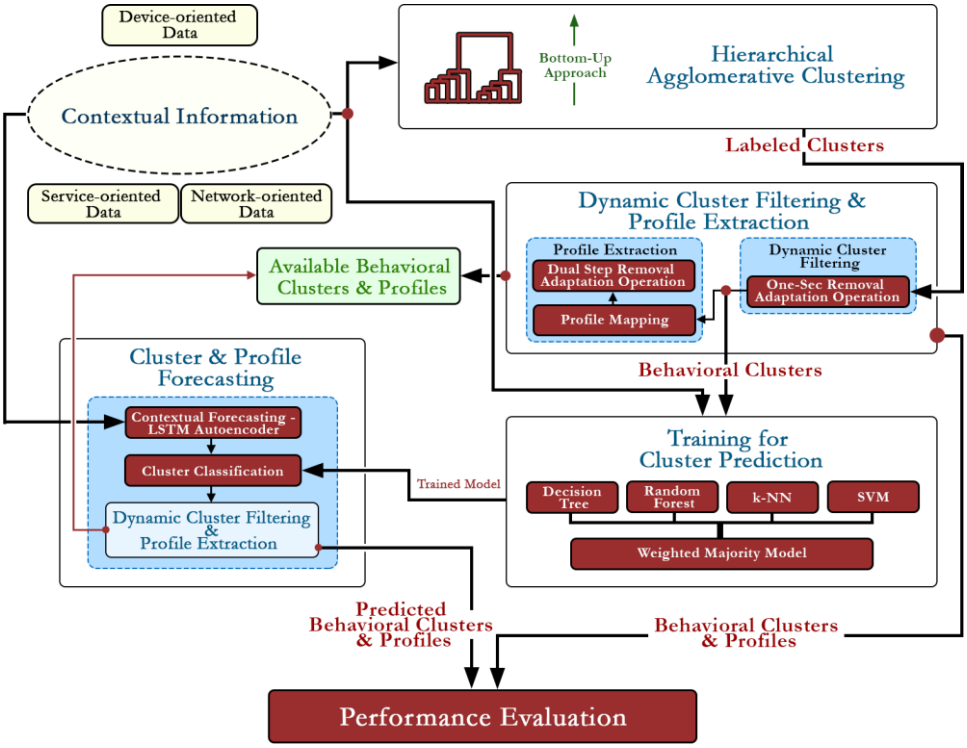


FIGURE 41: OVERVIEW OF DPM'S FRAMEWORK

Aggregation and pre-processing of Contextual Information

This module is responsible for aggregating and pre-processing all the available contextual information from the different network entities and segments as well as the logging modules. Additionally, in this step any potential irrelevant and/or redundant information present in the data set is removed.

Hierarchical Agglomerative Clustering (HAC)

This module exploits the output of the pre-processing module, in order to group similar data points (data features of the 3 different data entities) in common clusters named *Labelled Clusters*. In order to successfully extract the set of *Labelled Clusters*, a well-known unsupervised clustering method is

used, namely Hierarchical Agglomerative Clustering (HAC). This method provides great flexibility, since having a pre-defined number of clusters is not a requirement. Additionally, HACs results are reproducible. A measure of dissimilarity (distance function, linkage criterion) between sets of observations is required, in order to decide, which clusters should be combined.

The HAC method was implemented for each one of the 3 data entities separately. At this stage, a grid search process is introduced that calculates the appropriate number of clusters that best categorises the data features. The grid search process exhaustively searches for a combination of a measure of dissimilarity along with a number of clusters, that best satisfies its objective. The objective of the grid search process is to maximise the *Silhouette Value,* i.e., a metric that measures how similar a data feature is to its own cluster compared to other clusters, ranging from -1 to +1. A high silhouette value indicates that the object is well matched to its own cluster. The final output of HAC is 3 labelled data sets (one data set per entity), in which all data features are mapped to *Labelled Clusters*.

Dynamic Cluster Filtering & Profile Extraction

This module implements *Dynamic Cluster Filtering* and *Profile Extraction* mechanisms. The purpose of the first mechanism, is to remove the *Labeled Clusters*, which are considered to be outliers, from each one of the 3 data entities. We consider as outliers, the Labeled Clusters with duration of 1 second or less. Each outlier is replaced by a *Labeled Cluster* that last appeared for more than 2 seconds. This replacement operation is named *One-Sec Removal Adaptation*. An indicative example is given in Figure 42. The final output of this mechanism is the filtered set of *Labelled Clusters*, namely the *Behavioural Clusters*.

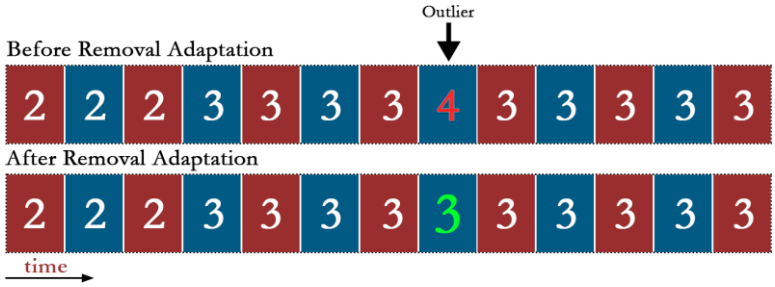


FIGURE 42: EXAMPLE OF ONE-SEC REMOVAL ADAPTATION OPERATION

The second mechanism is responsible for a 2-fold operation. The first one is called *Profile Mapping* and is responsible for assigning labels to the *Behavioural Clusters*. This is achieved by mapping unique triplets into single unique labels. One triplet consists of three *Behavioural Clusters*, one for each of the 3 data entities. Each one of these unique labels is defined as a *Behavioural Profile*. An indicative example is given in Figure 43.

	Behavioral Clusters			Behavioral Profiles
	Device	Service	Network	Profile
time ↓	CL_1	CL_1	CL_3	PR_1
	CL_1	CL_1	CL_3	PR_1
	CL_1	CL_2	CL_1	PR_2
	CL_1	CL_2	CL_1	PR_2

FIGURE 43: EXAMPLE OF PROFILE MAPPING OPERATION

The second operation of *Profile Extraction* is called *Dual Step Removal Adaptation* and is responsible for the removal of *Behavioural Profiles*, which are considered to be outliers. These outlier *Behavioural Profiles* are considered to be falsely extracted and thus they need to be replaced. The replacement method follows the same logic as the one presented for the *One-Sec Removal Adaptation*. This operation adds a second layer of filtering, after the first stage outlier filtering. The removal of the falsely extracted profiles may lead to additional reduction of the final *Behavioural Profiles* and as a result further enhance the quality of the output. Intuitively, the improvement in quality comes from the fact that fewer profiles with longer duration provide easier monitoring and management. The set of *Behavioural Clusters & Profiles* is being stored in a data pool named *Available Behavioural Clusters & Profiles* in order to be used as a mapping guide from the *Cluster & Profile Forecasting* module (See below "Cluster & Profile Forecasting" paragraph).

Training for Cluster Prediction

This module exploits the extracted *Behavioural Clusters* in conjunction with the initial contextual information, in order to train an ensemble *Weighted Voting Model*. This model combines the predictions from 4 different well-known supervised ML models (called base models), namely a Decision Tree (DTR), a Random Forest (RF), a k-Nearest Neighbours (k-NN) and a Support Vector Machine (SVM). The result of this module is the classification of newly introduced contextual information by predicting the respective *Behavioural Clusters*. The design decision to predict *clusters* -instead of profiles- was made in order to provide the possibility to discover newly-discovered triplets of clusters, -and as a result new profiles (through the process of *Profile Mapping*); respectively, the prediction of profiles would limit the prediction space, as the number of profiles would be static.

The heavy task of training the model is performed offline using the already collected contextual information. Having trained the model offline, the prediction process decreases in execution time, making it suitable for near real-time operation. The development of this model is of high importance, as it provides the capability of exploiting newly added or forecasted contextual information in a supervised manner.

Cluster & Profile Forecasting

This module is developed to perform cluster and profile forecasting for predefined time windows. More specifically, the module attempts to forecast contextual information that will be provided as input to the already trained ensemble model, introduced in the previous step, in order to predict the respective clusters. These clusters are exploited by the *Dynamic Cluster Filtering & Profile Extraction* module which is capable of extracting the final forecasted *Behavioural Clusters* and *Profiles*. This module comprises two different operations. The first operation, named *Contextual Forecasting - LSTM*

Autoencoder relates to the forecasting of contextual information for each one of the 3 data entities. The forecasting model is a Long Short-Term Memory (LSTM) Autoencoder model for *multivariate multi-step time series* data. The core functionality of this model is to forecast contextual information on multiple time-steps into the future (*multi-step*), using multiple prior time-steps of contextual information (historical contextual information). The selected LSTM Autoencoder is an implementation of an Encoder-Decoder LSTM architecture and is designed to efficiently perform a sequence-to-sequence (seq2seq) prediction.

The second operation of the *Cluster & Profile Forecasting* mechanism is called *Cluster Classification*. This operation is responsible for exploiting the forecasted contextual information in conjunction with the exported *Weighted Voting* trained model from the previous step (cf. above "Training for Cluster Prediction" paragraph) in order to perform cluster classification. The output of the model is the extracted clusters for each one of the 3 entities. This set of extracted clusters may also contain outliers that need to be filtered. The output set of this operation are the *Predicted Behavioural Clusters*. These *Predicted Behavioural Clusters* should be mapped into the final profiles. In order to extract the final profiles we initially map the filtered clusters into profiles and then add a second layer of filtering to remove the potential outlier profiles. This procedure is once again performed by applying the *Profile Extraction* mechanism introduced in the second step of DPM (cf. above "Dynamic Cluster Filtering & Profile Extraction" paragraph). In this case, the mechanism is using the *Available Behavioural Clusters & Profiles* pool as a mapping guide. The final extracted profiles of this mechanism are denoted as *Predicted Behavioural Profiles*. It is worth mentioning that it is possible for the *One-Sec Removal Adaptation* operation to output triplets of *Predicted Behavioural Clusters* that were not previously identified. As a result, the *Profile Extraction* mechanism will extract newly introduced *Predicted Behavioural Profiles*.

Performance Evaluation Module

This is the final module of the DPM framework. It is responsible for collecting the predicted (*Cluster & Profile Forecasting module*) along with the ground truth (*Profile Extraction & Dynamic Cluster Filtering*) behavioural clusters and profiles in order to evaluate the performance of the forecasting module. The corresponding performance evaluation outcomes can be found in Section 4.9.3.

3.2.1.3.4 Integration with the 5Growth Stack

In order for the DPM to be successfully integrated with the 5Growth stack, the 5Gr-AIMLP should be able to provide through the *Interface Manager* the already trained models used by the *HAC* module, the *Training for Cluster Prediction* module, as well as the *Cluster & Profile Forecasting* module. The DPM is flexible enough to adapt its data model to the available data provided by the Monitoring Platform. The complementary functionalities introduced by the DPM, namely One-Sec and Dual step removal and profile mapping operations are executed in the 5Gr-SO (or 5Gr-RL). The final DPM output, namely the Behavioural Profiles could be further exploited by the respective resource management/service orchestration mechanisms of the 5Gr-SO (or 5Gr-RL).

3.2.1.3.5 Innovation contribution to cover gaps

SO self-adaptation actions

The DPM is capable of analyzing and extracting profiles for 3 data entities (Network, Service, Device), providing information concerning network-, service-, as well as UE-related behaviour (e.g., device mobility patterns, service consumption). As a result, this extracted knowledge could be exploited by the 5Gr-SO towards proactive self-adaptation, generating considerable gains for the network.

SO automatic network service management

The DPM is able to monitor and process data aggregated from diverse network entities and segments, as well as logging modules, and to dynamically generate service-related behavioural profiles. The latter is based on historical information regarding service consumption and respective resource requirements. The output of the DPM could potentially be exploited by the 5Gr-SO towards automatic network service management.

3.2.2 Innovation 9 (I9): Anomaly detection algorithms

Due to the heterogeneity of the services that 5G and beyond network environments will support, the current innovation proposes an AI-based module in order to assist administrators and slice tenants to predict and diagnose anomalies among the services of the different slices deployed on the virtualized infrastructure. The focus of 5Growth Innovation 9 is on the prediction of network related anomalies. The proposed AI framework is capable of analysing aggregated and fine-grained data, such as resource-level data, RAN measurements, service KPIs, as well as traffic and mobility patterns, in order to predict potential network anomalies; therefore, the algorithmic framework will be presented along with its contributions to cover gaps, which are summarized at the end this section. The corresponding performance evaluation outcomes can be found in Section 4.10.

3.2.2.1 Algorithm Description

The proposed framework comprises three main phases which are illustrated in Figure 44. The first phase involves the pre-processing of the input data, along with the feature extraction step. Table 10 illustrates the features of the training dataset. It should be noted that the pre-processing of the features includes a min-max normalization, transforming every value in the $[0,1]$ interval, so that each feature contributes in a proportional manner to the ML models that will be applied.

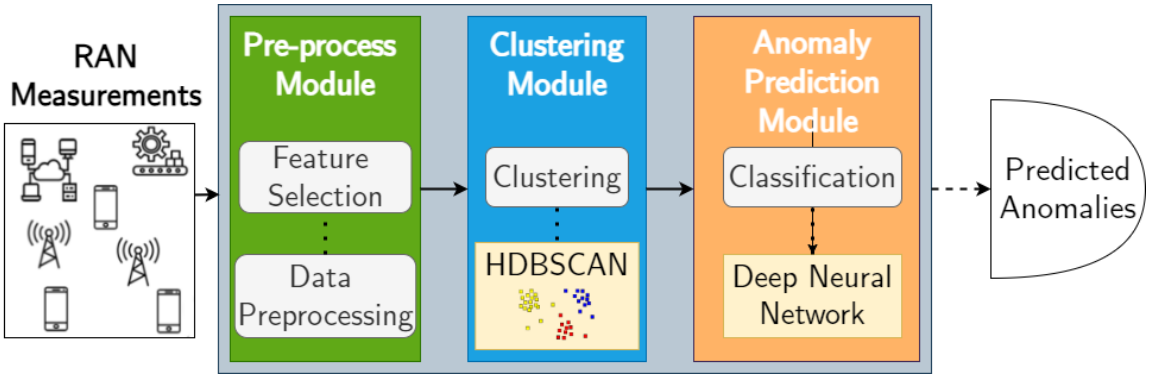


FIGURE 44: ANOMALY DETECTION ALGORITHMIC FRAMEWORK

TABLE 10: NETWORK KPIS USED AS FEATURES FOR THE TRAINING SET

Feature Name	Description
lost_packet	# of lost packets, per service per user
ul_delay	Uplink delay (ms)
dl_delay	Downlink delay (ms)
rsrp	RSRP (dB)
transfer_protocol	TCP of UDP encoded [0,1]
ulrx_cell	UE bytes received from the cell

The next phase concerns the exploitation of a clustering method using an unsupervised machine learning clustering model, where outliers and abnormal patterns of the time series are grouped together. The applied algorithm used for this step is Hierarchical Density-Based Spatial Clustering (HDBSCAN) [53], which is an extension to Density-Based Spatial Clustering (DBSCAN) by converting it into a hierarchical clustering algorithm. HDBSCAN is a robust clustering approach for anomaly detection in time series, which can capture, as well as predict, clusters of varying densities. The number of clusters is determined beforehand and is based on how many severity levels a network administrator targets, thus increasing the granularity of the prediction. Therefore, by adjusting the *epsilon* value of the distance metric of the model, the number of clusters formed is also changed (e.g., a large distance value generates two clusters *[-1,1]*, which means that samples with assigned cluster 1 are normal and -1 anomalies). Figure 45 illustrates the training dataset's time series, along with its assigned cluster; three potential clusters are defined, namely [C1: normal, C2: moderate, C3: anomalous] behaviour. Three KPIs have been selected, namely *[lost_packets, ul_delay, dl_delay]*, to illustrate the validity of the clustering algorithm by comparing the variance of the selected KPIs with the assigned cluster, for each time-step.

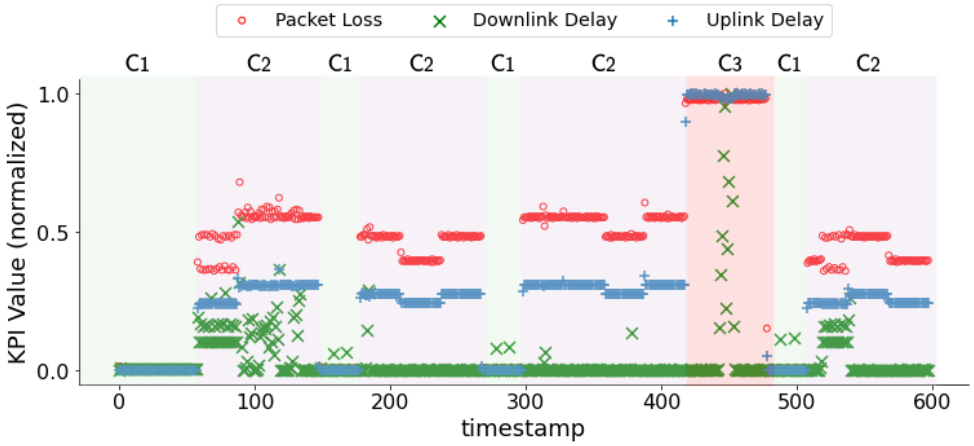


FIGURE 45: INDICATIVE NETWORK KPIS FOR CLUSTERING EVALUATION

Finally, the last phase of the proposed framework includes a feed-forward Deep Neural Network (DNN), which is exploited to perform a classification task, with the training dataset as input, along with each cluster as the Label/Target value for each sample, as assigned by the previous phase. More specifically, the DNN model consists of 5 layers X→64→64→32→Y, where X is the number of features in each sample and Y the number of clusters determined by the clustering algorithm (one-hot encoded), Rectified Linear Unit (ReLU) activation function, and categorical cross-entropy as a loss function. The aforementioned classification scheme is then used to assign a cluster to new and unseen data in real-time and hence predict potential system anomalies.

3.2.2.2 Integration with the 5Growth Stack

A mechanism that allows the anomaly detection module to dynamically ingest new data from the 5Gr-VoMS in a similar way to what was explained in previous innovations. These data consist of RAN/Edge-related network measurements that are produced by the network functions involved in the service and consumed by the anomaly detection module. Once the proposed anomaly detection algorithm predicts an anomaly in the network, an anomaly notification is produced and later consumed and exploited by any 5Gr-entity in real-time. Addtionally, the dataset, as well as any parameters and metadata used for the offline training of the Machine Learning Model can be provided to the 5Gr-AIMLP (I5) through an API.

3.2.2.3 Innovation contribution to cover gaps

SO automatic network service management

Due to the heterogeneity of the NFV-NSs that the 5Growth environment supports, robust network service management is required, including flexible, automated SO service/VNF migration, scaling out/in, etc., particularly for cases, in which one or more network components may demonstrate unusual or abnormal behaviour. Such cases of abnormal behaviour are identified by the Anomaly Detection (ADT) module, that exploits a robust clustering approach for anomaly detection in time series, and which can capture, as well as predict, clusters of varying densities and types. The feed-

forward Deep Neural Network (DNN) that is afterwards exploited for classification (ADT's second sub-module), uses the aforementioned clusters, in order to process unseen data in real-time and hence detect potential system anomalies, node/link failures, etc. Those types of anomalies could potentially lead to SLA violation and the proactive anomaly forecasting could be exploited by the 5Gr-SO in terms of automatic network service management by applying self-adaptation actions.

SO self-adaptation actions

The anomalies, predicted by the ADT module, could be pushed in the form of alert events, to be consumed by any "subscribed" network module, such as the 5Gr-SO.

SO dynamic monitoring orchestration

As the network measurements are gathered by monitor functions provided by the Monitoring Platform, according to the way in which this Monitoring Platform has been designed, it implies the control and orchestration of the related modules of the 5Gr-VoMS to enable the monitoring of the network metrics proposed, enabling consequently the dynamic monitoring orchestration of these functions supporting the anomaly detection algorithm proposed in this Section.

RAN support

The AI-driven ADT framework is capable of analyzing RAN/Edge network measurements in order to identify, but not limited to, potential RAN-related abnormal behaviours. The latter could be pushed in the form of alerts to any internal/external module and could be exploited towards supporting and configuring the RAN infrastructure.

3.2.3 Innovation 10 (I10): Forecasting and inference

Novel 5G use cases typically have strict network requirements (e.g., availability or minimum guaranteed bandwidth). Meeting such requirements is not only a matter of using powerful and recent hardware, but a proper mapping between users' demand and associated resources. Currently, with increased utilisation of Virtualised Network Functions (VNFs) that are sharing common hardware resources, mapping between users' demand and associated resources quickly is becoming more important. However, user demands might vary as a function of time. As an example, the reader can picture a collision avoidance service with Multi-access Edge Computing (MEC) servers co-located with radio base stations along a highway. During rush hours the V2N services are very likely to foresee bursts of demand that cannot be accommodated by the turned-on MEC servers along the highway. In this case, a typical approach to dimension resources is to allocate them based on the peak demand. However, such approach might conduct to high inefficiencies, especially if demand variations are high and resources are shared by many services. Forecasting arises as a candidate solution to predict how demand will evolve over time, allowing better and more efficient resource sharing and utilization and, thus, avoiding or minimizing SLA violation by the offered services.

Predictive techniques have been widely used to forecast road traffic in specific locations. On one hand, the use of classical time series analysis to forecast road traffic flows started to be studied, with methods such as Error Trend and Seasonality (ETS) forecast, Autoregressive Integrated Moving

Average (ARIMA) and Triple Exponential Smoothing (i.e., Holt-Winters) [54]. On the other hand, with the emergence of AI/ML techniques, their applicability to forecast road traffic flows also started to be tackled by the research community. The works in [55][56] appear as the first to apply, respectively, Stacked AutoEncoders (SAEs) and Restricted Boltzmann Machine (RBM) models to forecast road traffic flows. In [57], a deep regression model with four layers (including one input, two hidden and one output layer) is used to forecast vehicle flows in a city. In turn, the utilization of LSTM is also leveraged in [58]. Other techniques were also applied, like Deep Belief Network (DBN) in [59], Dynamic Fuzzy Neural Networks (D-FNN) in [60] and Gated Recurrent Units (GRU) in [61].

The considered forecasting techniques are: Double Exponential Smoothing (DES), Triple Exponential Smoothing (TES), Hierarchical Temporal Memory (HTM), Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), Temporal Convolutional Networks (TCNs), Convolutional LSTM. DES and TES are forecasting technique based on time series analysis. Hierarchical Temporal Memory (HTM) is a machine learning technology that aims to capture the structural and algorithmic properties of the neocortex. The core component of the HTM forecaster is a temporal memory m_t consisting of a two-dimensional array of cells that can either be switched on or off and that evolves as the timestamp t increases (t = 0, 1, 2, . . .). LSTM is a special form of Recurrent Neural Network (RNN) that can learn long-term dependencies based on the information remembered in previous steps of the learning process. Gated Recurrent Units (GRUs) are neurons used in RNNs, and as LSTMs cells, they store a hidden state that is recurrently fed into the neuron upon each invocation. The usage of TCNs in this work relates to the results reported in [57], where authors claimed that they implemented a traffic jam forecasting methodology using a deep learning infrastructure with two hidden layers, and convolutional neural networks. In the convolutional LSTM (TCNLSTM), both TCN and LSTM models are combined into a single unified framework and trained together as described in [62].

3.2.3.1 Innovation contribution to cover gaps

SO automatic network service management and SO self-adaptation actions

Zero touch Network and Service Management is one of the ETSI standardisation activities toward a closed-loop management of network and services [63].

Different services and network segments might present some specific requirements about the closed-loop response time that is the time it takes from the network state change to the network adaptation. Forecasting can represent a data analytics technique that could help to shorten this response time. Indeed, by forecasting future changes the system could proactively take countermeasures and initiate network adaptation.

3.3 Framework Innovations

This section describes those innovations that are transversal and help build the framework of the 5Growth project.

3.3.1 Innovation 11 (I11): Security and auditability

The tighter integration of business verticals into network operations, as in the 5G Vertical Slicing use-cases, raises significant security, trustworthiness, and reliability issues [64][65][66]. A single business vertical must not compromise the integrity of the platform that powers all other verticals. Similarly, the vertical applications that are entrusted to the platform should not be compromised by external parties. Cloud Computing, and by extent Edge Computing, encourage a de-facto standardization of the same protocol stack (TCP/IP) [67] across previously segregated systems. The use of standardized protocols, and the reuse of similar software solutions, substantially lowers the cost of an attack. One vulnerability within a component of this stack now covers a broader range of potential targets.

To achieve a successful attack and exploitation of a target, the attacker must first do a reconnaissance of the virtual surroundings, identifying the objectives, along with the vulnerabilities of each service that may lead to its goal [68]. Security vulnerabilities are mostly a particular kind of software bug [70], and bugs tend to be revealed over time. Therefore, time is on the attackers' side to find and exploit a vulnerability, while network function maintainers have time playing against their security designs; this effect of time is known in the MTD field as the asymmetric relationship between attackers and defenders. The core principle of MTD is to identify the available network configuration space and dynamically explore it (i.e., create movement) to frustrate the attackers' endeavors [71][72]. Provided the attacker does not know the secret that creates the movement, the MTD mechanism may break the attacker's connectivity to the service or forge data (or interactions) so that the attacker acts upon faulty intelligence.

The practical issue with MTD is how to generate movement across peers without disrupting legitimate usage, hindering other ancillary systems (such as auditing, accounting, and billing), or being detrimental to an attack's attribution and forensics [73]. Synchronizing the movement across peers without incurring heavy overheads (synchronization protocol), sacrificing mutation speed, or connection reliability (i.e., errors due to desynchronization) is a significant challenge. Furthermore, the intended scope requires adding/removing peers at any time without disrupting the service or compromising the key management practices.

Our solution is inspired by the HMAC-based authentication methods already widely deployed by other network services. On the one hand, those HMAC-based authentication methods do not replace the existing account authentication mechanisms but complement them with an extra factor. The MTD approach has similar nature, adding an extra factor to the existing defense mechanisms (e.g., firewalls, Intrusion Detection Systems (IDSs), or Intrusion Protection Systems (IPSs)). On the other hand, the HMAC-based authentication methods in other network services already established practices that deal with peer synchronization, peer management, and key management issues.

If the attacker can interact with the network socket of sensitive services, then time is on its side to find potential vulnerabilities in that service stack. For example, a remotely triggered Control-Flow hijacking vulnerability [76] could allow running arbitrary commands with elevated system permissions, thus subverting the application logic and bypassing the security controls in place. A prime example of this vulnerability class is the EternalBlue [69] exploit that subverted Microsoft

Windows' SMB service. To curb the asymmetric relationship between the attacker and the defender [71][72], we propose a port-mutation MTD mechanism designed to comply with the performance, management, and scalability requirements of sensitive network functions in Edge computing, 5G, and Network Slicing. The proposed mechanism works alongside other existing controls, effectively acting as a network-bound seamless HMAC-based authentication code.

The existing HMAC-based authentication solutions deployed by the other network services already deal with peer synchronization and management. They do so using the device clock as a seed for the authentication code, alongside the shared secret and the negotiated parameters (such as code size and expiration period). Because the synchronization is achieved using the universal time, new peers can be added/removed at any instant without disrupting the synchronization with other peers and their ability to authenticate into the system. No active communication is required between the peers to achieve that synchronization (i.e., no additional overhead, and the new peer is immediately ready to authenticate). Isolated packet-loss incidents do not lead to permanent desynchronization. Furthermore, each authentication code has a hard expiration time, limiting an interception attack's usefulness to that validity window.

Our mechanism uses a slightly adapted authentication function to mutate the service ports, maintaining the familiarity and proofs of a widely known solution. However, using the authentication generator as the mutation function comes with its own set of challenges. We present in this document the most significant challenges and design aspects of our MTD mechanism. The complete considerations are in the supporting article [96].

We have addressed the lowered bit-strength of the authentication code by generating new codes more often, going for expiration intervals in the tens of milliseconds instead of minutes. However, making the authentication codes that much shorter-lived has a substantial impact on the systems' performance and synchronization, vital to making such a mechanism work (at all). The clock source needs to be synchronized up to the required precision of the mutation, which is achievable with readily available solutions such as GPS time [77], NTP [78], or PTP – IEEE 1588 [79]. We have verified that NTP is sufficient to achieve the required synchronization.

Our solution consists of an SDN application (the MTD Authenticator) that can either work with physical switches or be packaged together with an internal virtual switch inside a VNF (as shown in Figure 46-a). Being a VNF within the ETSI NFV platform crucially allows for the on-demand instantiation and scaling of fully virtualized systems. Crucially, because the IP address is the primary identifier of the peer selection, our horizontal scaling strategy consists of Policy-Based Routing (PBR) to load-balance the peers across the available instances. The PBR approach allows to dynamically redirect the load as the VNF scales in/out. Our design also targets the NFV-MANO primitives to provide the required interfaces to add/remove peers across each instance of the solution, fundamental for flexible orchestration and enabling horizontal scalability. Furthermore, the design also allows for forwarding plane hardware offloading to physical network equipment (such as SDN-controlled switches) to make intelligent use of existing resources during the most computationally demanding phase (i.e., per-packet authentication enforcement). We will further detail the inner workings next.

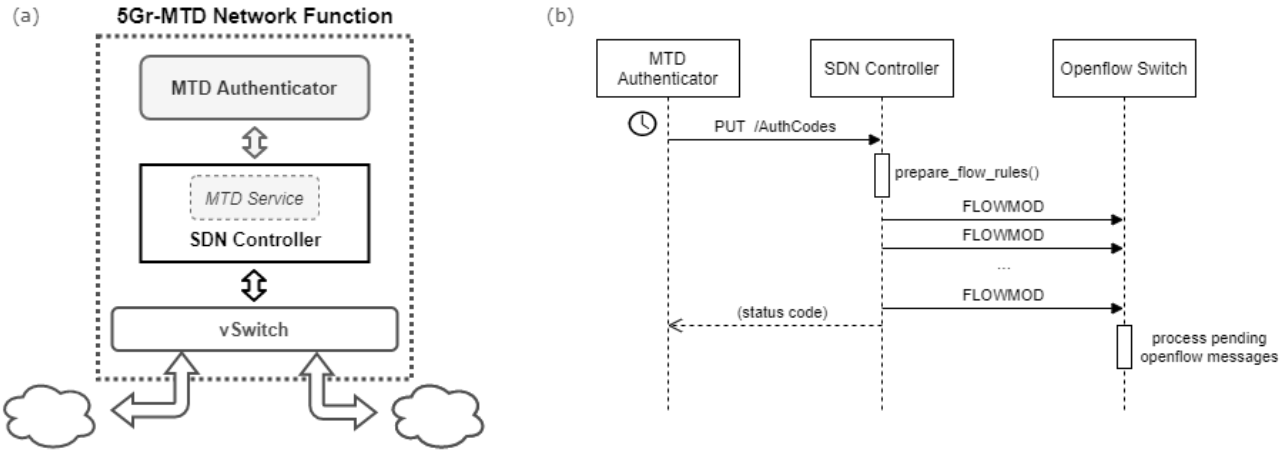


FIGURE 46: (A) PROPOSED NF, (B) CONTROL-PLANE SEQUENCE INSIDE THE NF

The 5Gr-MTD NF relies heavily on SDN-controlled switches to achieve the stringent forwarding-plane performance requirements of edge computing, 5G, and Network Slicing. Enforcing the authentication on a per-packet basis would be a very demanding task without a performant datapath, as shown by the prior work. We have specifically chosen an existing field in the TCP/UDP protocol header (the source/destination port) to send the authentication codes, thus minimizing the data overhead and crucially allow offloading this task to an SDN switch. This offloading enables more flexible approaches than purpose-built implementations of a 5Gr-MTD datapath: SDN switches are readily available and can be used for other functionalities than just our 5Gr-MTD function. However, SDN switches are not designed to calculate cryptographic functions as part of their packet matching rules (i.e., they cannot calculate the authentication code). The code calculation must be done in the control plane to make full use of standard equipment via an SDN Application (the MTD Authenticator, as shown in Figure 46-a).

The Authenticator will calculate the valid code for each time slot (according to the secret shared among the parties) and post the updated codes to the SDN controller at the right time instances (as shown in Figure 46-b). Having the SDN application controlling the timings is a crucial element of the design. It simplifies the controller's logic, allows for different expiration times for different peers, and enables future work towards a more advanced re-synchronization and delay compensation (e.g., using more data sources).

There is an MTD service built into the controller, serving a dual purpose. The first is minimizing the number of messages in the northbound and their respective sizes compared with issuing requests for many FLOWMODs. The second was handling the PACKET_IN events when the received code does not match, mainly gathering relevant network data (e.g., topology data) and the packet's timestamp before reporting the anomaly to the threat decision function. When packaged together as a VNF (as shown in Figure 46-a), the solution enjoys full on-demand instantiation and scaling. When used alongside a hardware switch, it takes advantage of the efficient hardware offloading that purpose-built resources provide while still retaining the same virtualization benefits to the control-plane functions. The proposed MTD system requires at least two NF instances, one for each end of the connection.

In this solution, the peers are identified by their current endpoint configuration, particularly their IP addresses, as acquired from their respective Network Access Control (NAC). Both clients and servers can simultaneously connect to multiple endpoints, having different mutations (or no mutation) happening according to the protocol, port, and IP address match. The server shall not expose another service in the same listening IP address and protected protocol, under the penalty of having broken connections once a calculated authentication code overlaps with that service's port. Serving additional services using the protected protocol will require another IP address. There are no such caveats for the client.

Upon receiving the peer's authentication codes list from the Authenticator (Figure 46-b), the controller will then generate flow rules that enforce the valid codes and issue the required FLOWMODs to the switch. The Authenticator is then notified of the successful processing of the code list into flow rules (i.e., "200 OK"), and the forwarding plane will (eventually) start enforcing the new authentication codes sometime after the switch processes the FLOWMOD messages. Because time is of the essence in our solution, the rules generated need to account for packets that fall right at the limit of a time slot, some jitter in the network, or even slight clock skew/drift. We solved this using a strategy that considers valid a set number of adjacent authentication codes before/after the current time slot (e.g., [-1, 1] valid slots. The valid receiving authentication codes are updated in two stages to minimize transient states. First, we create a new OpenFlow table with the new authentication codes. Then, we update the rule in the default table so that the incoming packets are resubmitted to the new authentication codes table (this is a Nicira extension to the OpenFlow protocol). Lastly, we delete the table with the old authentication codes, hence clearing those expired codes before reusing the table in a later update.

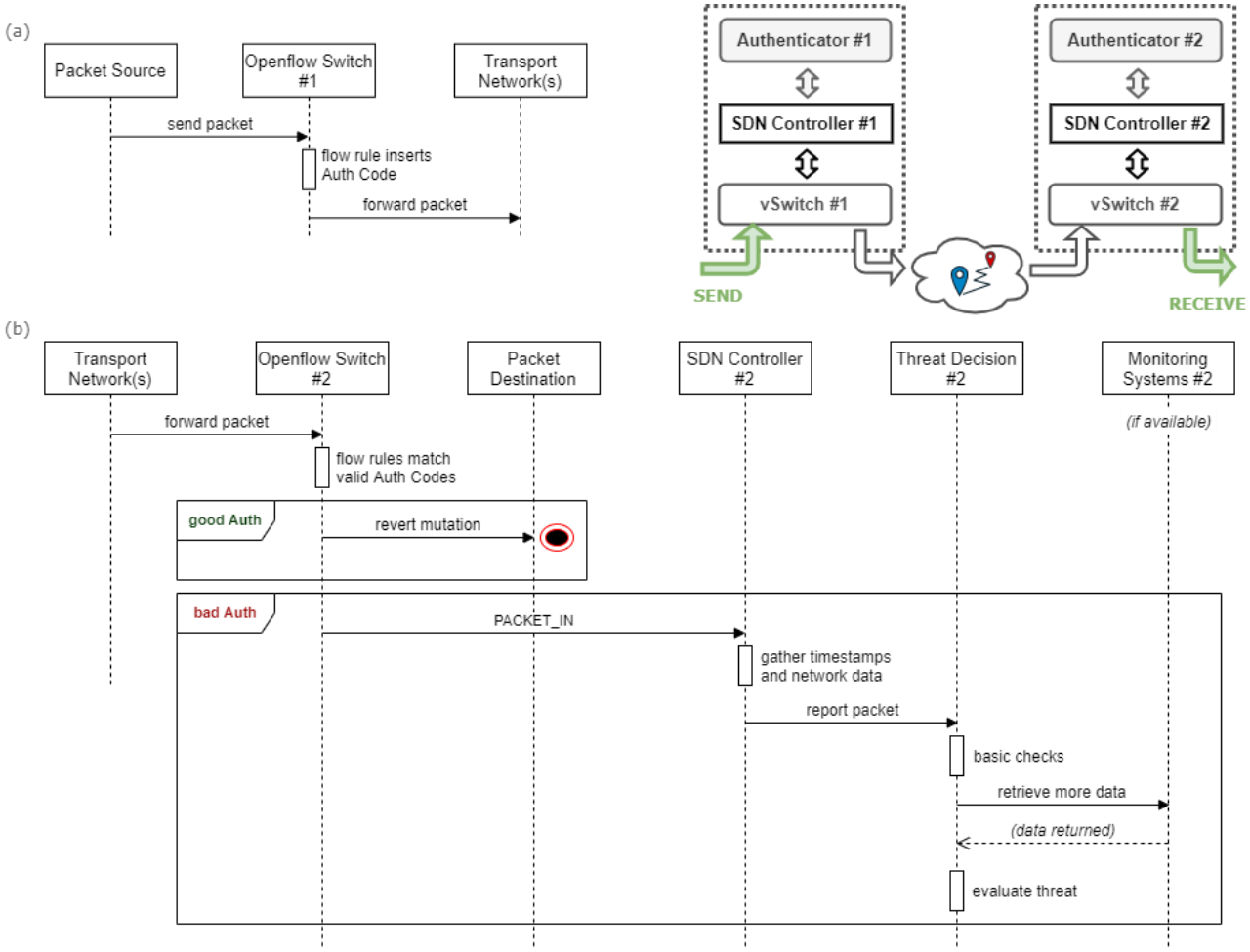


FIGURE 47: FORWARDING-PLANE SEQUENCE

Meanwhile, in the forwarding plane, the packets are being processed accordingly to the sequence shown in Figure 47. The figure illustrates the sequence taken when the packet goes from the sender to its destination. Each 5Gr-MTD NF's roles are numbered as shown in the top right corner (#1 for send, #2 for receive). The solution allows the packets to flow in both directions. It is just a matter of reversing the roles (and numbering) when receiving the response and following the appropriate sequence diagram. Because we can only have one destination (or source) port in the packet header, inserting the authentication code is an unequivocal transformation (Figure 47-a).

The certainty while sending is in stark contrast with receiving the packet (Figure 47-b), which may have multiple valid codes accordingly to the set grace period (e.g., [-1, 1] valid authentication codes). The same flow rule that matches a correct authentication code will also revert the mutation to the protected service's actual port. However, when the received code does not match, we need to assess if that mismatch is due to a threat or some other innocuous condition (e.g., network jitter). Our solution handles this mismatch through a regular PACKET_IN to the SDN controller, which will record the event's timestamp and other relevant network data (e.g., topology) and then send it to a threat decision service. The threat decision can have more data available from other monitoring systems of the hosting platform. That will help assess if there was any recorded delay (e.g., network, CPU, or others) that, once factored in, would yield a different received time slot in which that code was still

valid. That will eliminate false positives for the alarm generation system while still stopping the data plane threats. If the mismatch is not consistent with the regular platform operation, an alarm can be generated so that further analyses are performed to understand the severity of the threat. The balance between eliminating false positives and not creating too many false negatives is crucial for our solution because of commonly occurring events such as network jitter.

The intelligent use of monitoring, well-defined constraints, and the events gathered at the threat detection system would allow the Authenticator to account for the enforcement delays caused by the switch at either end of the communication. It may as well adjust the authentication code calculation according to the estimated end-to-end delays so that the code remains valid when the packet arrives at the destination (improving performance). As we will show in the evaluation Section 4.12, the solution is already workable for our scenarios and has a suitable performance even with just coarse monitoring data.

3.3.1.1 Innovation contribution to cover gaps

Integral Security

This innovation provides a cutting-edge take on security provisioning for control plane interactions by establishing greater underlying resilience against network-centric attacks. It is a new way to bring two-factor authentication into the dynamic network procedures into the cloud-based telco of 5G. It protects sensitive network services by using a port mutation akin to a seamless time-based one-time password authentication. It uses cloud-based enhancements such as NFV and SDN to perform port mutations. It can work exclusively as a virtual networking function with instantiation on-demand or optionally in conjunction with OpenFlow hardware-accelerated switches for more intelligent resource usage. As such, the same underlying tools that have made possible the remainder of 5Growth's innovations (NFV and SDN) have also been laid out for the provisioning of advanced security capabilities.

3.3.2 Innovation 12 (I12): 5Growth CI/CD

The 5Growth CI/CD system is a software toolchain and combination of practices targeted to automate and simplify the development and deployment procedures for the 5Growth project. The 5Growth CI/CD, also mentioned as "5Growth CI/CD and containerization" in D2.2, consists of two parts: Continuous Integration (CI) responsible for automated testing of a new code, and Continuous Deployment (CD) responsible for simple, repeatable, and reliable deployment.

The high-level CI workflow is represented in Figure 48.

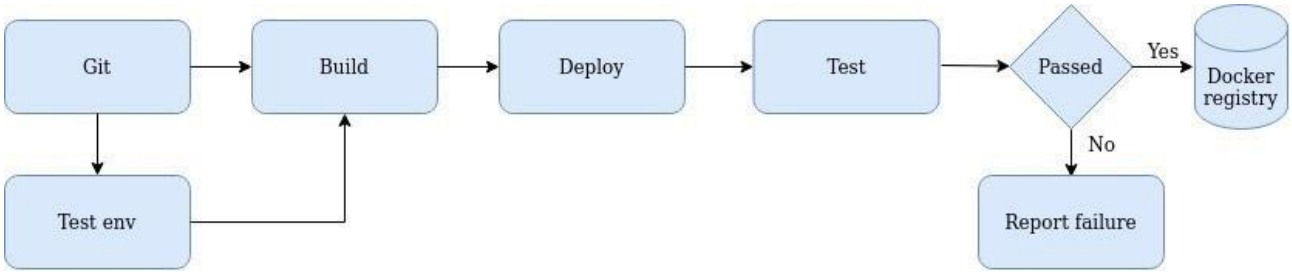


FIGURE 48: HIGH-LEVEL CI WORKFLOW

Newly committed code automatically triggers the verification pipeline. Firstly, the pipeline pulls the last change from the git repository, and then it creates the temporary test environment where the downloaded code is been built, deployed, and verified. The verification itself may include a set of automated tests, targeted to cover different aspects of the code quality assurance including integration testing, functional testing, and others. Depending on the verification result, the pipeline will either build needed artefacts or will finish the job by forming the failure report. Artefacts of the pipeline is a term describing the outcome of the code compilation, ready for being used for the deployment of the developed software. In this particular case artefacts would be docker images, that may be built, pushed to docker registry, and can be used for further deployments in case of a successful verification pass. The CI flow with test coverage can reduce the time for debugging in otherwise manually performed activities. This benefits the project and is considered the main goal of the 5Growth CI/CD activities.

The high-level CD workflow is described in Figure 49.

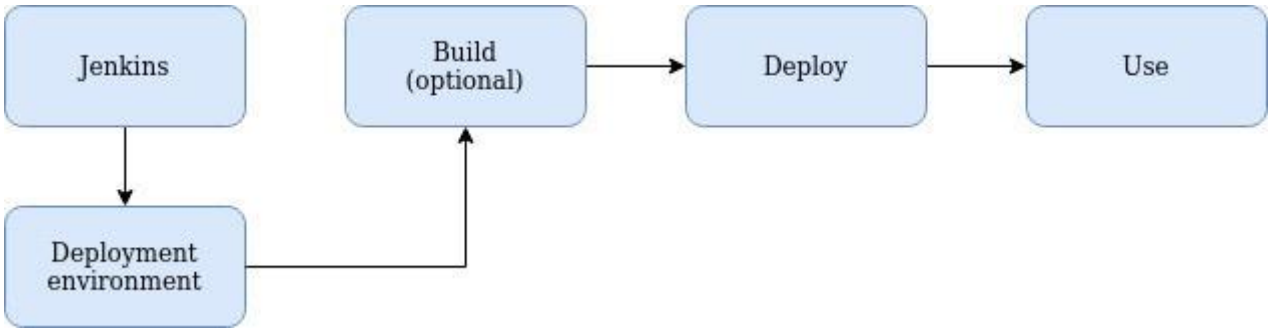


FIGURE 49: HIGH-LEVEL CD WORKFLOW

The CD workflow is highly customizable. The first option to choose is whether deployment will happen in the prearranged environment or instantiated by the CD pipeline. Then the user can choose parameters of the instantiated environment, like RAM amount, vCPU count, networking, etc. Then deploy the 5Growth platform on the chosen environment, using the general version for experiments or a dedicated pilot branch with customized configuration. 5Growth platform containerization was developed and implemented as a part of the 5Growth CI/CD initiative. This approach allowed to pack 5Growth platform in containers instead of software packages and go out of the virtual machine platform towards a container-based approach. This allows the 5Growth stack to become a cloud-ready application. During automation deployment development for CI/CD, there were developed artifacts that include automation scripts, orchestrator instructions, etc. that can be used outside of

the 5TONIC environment for 5Growth platform automated deployment. This allows the shipping of the platform as a whole. The 5Growth CI/CD architecture is depicted in Figure 50.

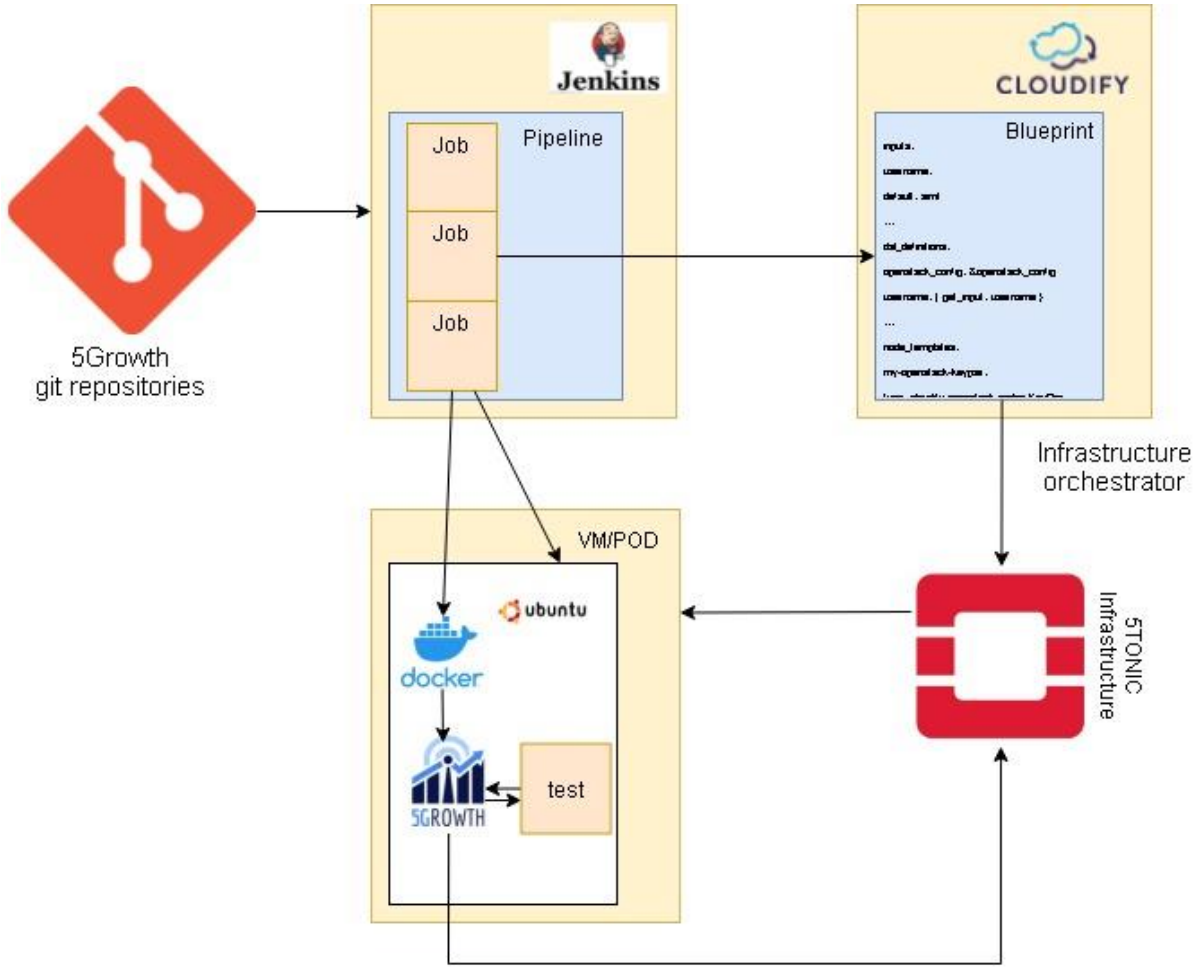


FIGURE 50: IMPROVED 5GROWTH CI/CD ARCHITECTURE

- 5Growth git repositories: This is a storage for the project codebase and a version control system, that allows project developers to contribute and collaborate. Jenkins continuously monitors 5Growth repositories for code changes and when it happens, the pipeline is being triggered.
- CI/CD tool: Jenkins2 has a set of preconfigured Jobs united in a Pipeline, providing automation for stages in a flow such as environment preparation, verification, containerization, etc. The pipeline is responsible for requesting test environment instantiation via an infrastructure orchestrator. For this, the dedicated Jenkins Job generates an orchestrator blueprint and passes it to the orchestrator.
- Infrastructure orchestrator: it instantiates environment according to provided blueprint, received from the Pipeline, monitors instantiation progress and reports back when the environment is ready. Orchestrator's blueprint connects the test environment to a network with the dynamically assigned IP address. The assigned address should be passed to the

Jenkins Pipeline to use this environment after instantiation. The orchestrator is also responsible for providing this address to the Pipeline.

The infrastructure orchestrator used for instantiating test environments on the OpenStack is Cloudify. Using the same orchestrator for CI/CD and 5Growth platform allows reducing resource usage. For 5Growth platform deployment on Kubernetes no orchestration is needed because Kubernetes is an orchestrator for containers and the environment can be described with the K8s primitives' configuration. When the environment is ready, the pipeline takes control and install prerequisites for platform deployment and deploy containerized 5Growth platform.

- Series of tests examine the deployed platform and generate a report. The pipeline analyses test reports and makes a decision. Successfully verified code changes are being built, containerized, and stored for next usage, while failed changes are reported to the developers.
- Infrastructure itself is responsible for hosting test, demo, and development environments, CI/CD infrastructure, and 5Growth platform payload.

In the scope of the innovation, the 5Growth CI/CD was extended in the following ways:

In containerization area:

- Orchestration extension to 5Growth platform deployment on Kubernetes.
- Automation for Kubernetes deployment in CI/CD pipeline.

To extend the containerized platform deployment stability and reliability there were deployed Kubernetes environment for hosting the 5Growth platform. To automate 5growth platform deployment on Kubernetes there were pipelines developed. These pipelines allow simplifying and automating the deployment of the 5growth platform into a Kubernetes environment.

Environment area extension:

- Developer's environment unification.
- IaC (Infrastructure as code) environment preparation for development, testing, and pilots (if applicable).

The CI/CD performs 5Growth platform build, deployment, and test regularly. To automate and speed up these processes there was the need to make infrastructure instantiation a part of the CI/CD Pipeline. Jenkins Job that allows instantiating required infrastructure was developed and integrated into CI/CD pipeline. This Job is used in testing and deployment pipelines, and thus all the nodes instantiated by a Pipeline are the same. This approach brings repeatability and reusability to the project.

Software testing area:

- Test coverage extension (automated testing pipelines extended with integration testing).

To increase the efficiency of the 5Growth CI/CD platform, it is important to verify interactions between the different 5Growth components. To cover this point there were different tests developed to verify deployment up and running or verify interactions between 5Growth platform components.

Thus, the 5Growth CI/CD features include the following:

- Containerized 5Growth platform deployment automation.
- Containerized 5Growth platform orchestration via Kubernetes.
- IaC environments creation for development, testing, and deployment purposes.
- Automated Pipeline trigger.
- Automated environment creation.
- Automated containerized components build.
- Automated containerized deployment on the dedicated environment.
- Automated tests run on the new deployment.
- Automated notifications in case of failure.
- Automated environment deinstantiation.

The detailed CI workflow is shown in Figure 51.

Our automated CD pipeline allows decreasing deployment time. A set of measurements validating this claim are presented in in Section 4.13.

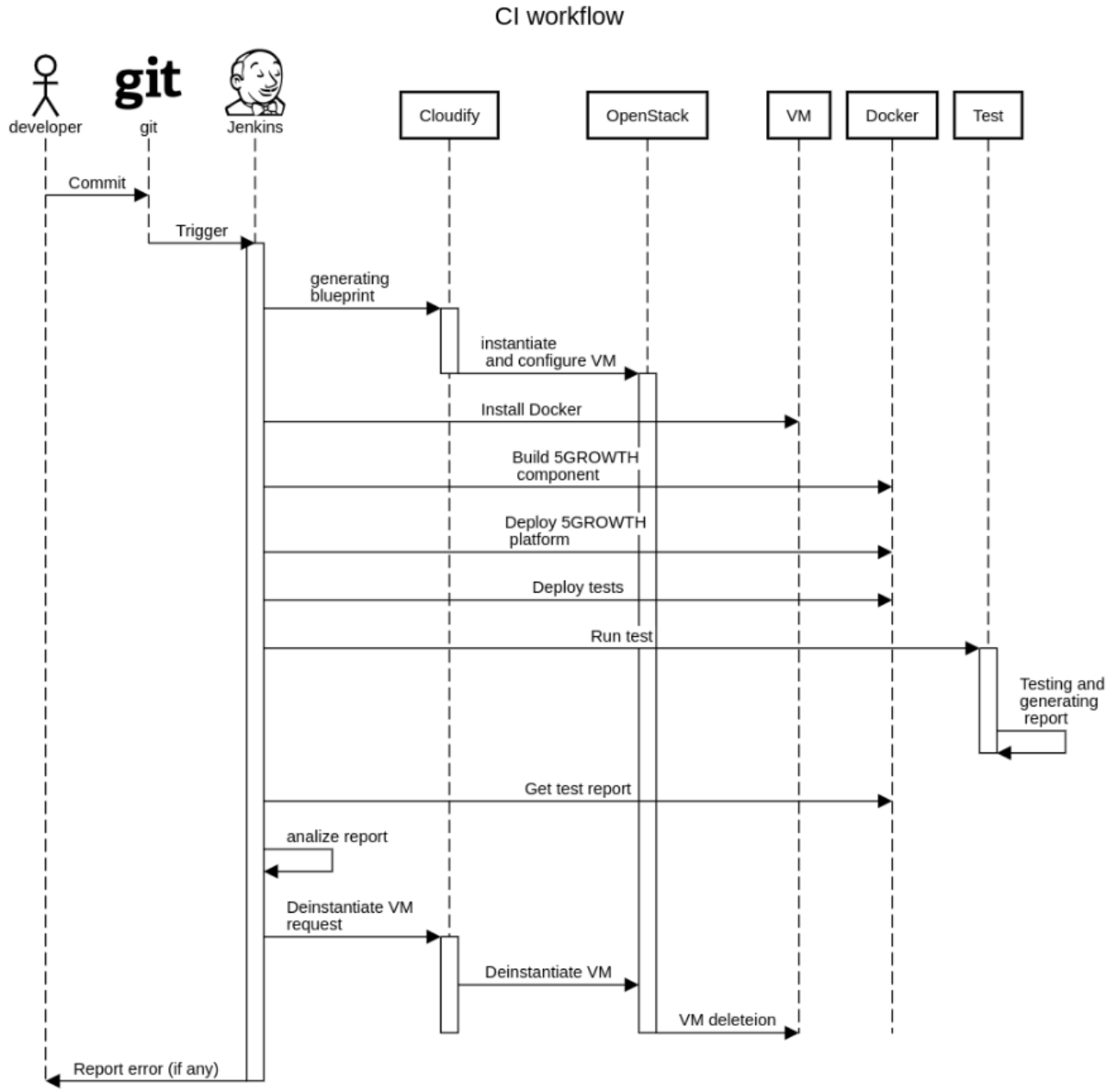


FIGURE 51: DETAILED CI WORKFLOW

3.3.2.1 Innovation contribution to cover gaps

SO automatic network services management

5Growth's CI/CD was developed as a framework under the umbrella of 5Growth project for 5Growth platform code validation and development automation. 5Growth CI/CD is developed for dedicated infrastructure usage, like the CI/CD tool itself, OpenStack, Kubernetes, Cloudify and hardware.

As described above, the 5Growth CI/CD framework consists of Continuous Integration (CI) features, responsible for automated testing of a new code; and Continuous Deployment (CD) features,

responsible for simple, repeatable, and reliable deployment and containerization, that allows to ship 5Growth platform as a whole.

Continuous development and integration

The main goal of 5Growth CI/CD is to contribute to 5Growth platform development through automation of development, integration, and deployment processes. To reach that goal on 5TONIC infrastructure, Jenkins, Cloudify, OpenStack, Kubernetes, and Docker registry infrastructure was deployed. All these tools tied together allow monitoring repositories, collecting changes there, test, built software, and containerizing them to be available to deploy any time. In this way, 5Growth CI/CD consists of two parts – Continuous Integration (CI) responsible for new code handling automation, described in Figure 48, and Continuous Deployment (CD) responsible for automated 5Growth platform deployment (Figure 49).

3.4 Conceptual Analysis of Innovations to support 5Growth pilot use cases

This section presents a conceptual analysis of how the feature gaps identified in D2.1 [1] and can be addressed by the innovations that solve them (Section 3) to fulfil the needs of the vertical pilots. In order to be as generic as possible, the analysis discusses about their possible general application to the varied 5Growth pilots and use cases. A summary is presented in Table 11.

TABLE 11: 5GROWTH PILOTS AND USE CASES

Pilot	Use case	Description
COMAU Industry 4.0	Digital Twin Apps	Digital reproduction of factory components to work on potential solutions to issues directly on the digital twin or to predict bottlenecks in the production/assembly line
	Telemetry/Monitoring Apps	Massive data collection of the factory floor to monitor and prevent failures of machineries
	Digital Tutorial and Remote Support	Tutorials and remote support to technicians and maintenance staff
EFACEC_E Energy	Advanced Monitoring and Maintenance Support for Secondary Substation MV/LV distribution substation	Assistance to remote operator and crews dispatched to the field to better assess the severity and the impact of an outage
	Advanced Critical Signal and Data Exchange across wide smart metering and measurement infrastructures	Validate advanced smart grid control solutions across wide smart metering and measurement infrastructure
EFACEC_S Transportation	Safety Critical Communications	Supporting data information exchange between the level crossing train detector activation points and the LX (Level Crossing) controller

	Non-safety Critical Communications	HD video image transmission from LX surveillance camera to the approaching trains' on-board tablets and/or maintenance agents' hand-held tablets, and Level crossing LX controller status and alarm events transmission to maintenance agents' tablets
INNOVALIA Industry 4.0	Connected Worker Remote Operation of Quality Equipment	Operator programs the coordinate measuring machine remotely using a virtual joystick in a mobile device, whilst a live video feed of the machine is displayed as feedback to the operator
	Connected Worker - Augmented ZDM Decision Support System (DSS)	Automatic guided vehicle and coordinate measuring machine control systems will be coordinated to share a single machine across multiple production lines

See D3.2 [80] for more details about the pilots and use cases.

3.4.1 Enhanced VS network slice sharing

In 5Growth the MANO stack was enhanced to enable the usage of AI/ML-based algorithms in the different management and control loops in a transparent manner. In particular, at the 5Gr-VS the control workflow, which is used internally to determine the network slice instances to be shared, has been updated to support the usage of external logic, which in 5Growth was implemented by an algorithm relying on AI/ML models. Moreover, these algorithms also determine if any scale-up or scale-down is to be performed on the shared slice in order to accommodate the incoming service. We refer to Section 3.1.5 for further details regarding the specific arbitration algorithm. The aim is to demonstrate this innovation in the COMAU Pilot, as further detailed in Section 3.4.2.

Network slice sharing in 5G-VINNI is natively supported by the SONATA Service Platform in terms of (re-)use/combination of already running nested-NSs. For this to be possible, NSDs have an option to mark each service as shareable or not shareable. This feature can be used by all the Efacec pilots,

3.4.2 VS arbitration at runtime

The 5Gr-AIMLP has not yet been integrated in any pilot; however, its benefits for smart factory use cases, such as those targeted by the COMAU pilot, have been demonstrated in the case of the Digital Twin application and are under further development within WP3.

A smart factory requires indeed the implementation of several services at the network edge where computing resources are limited, and most of such services have strict latency constraints. This calls for mechanisms that allow for an efficient usage and sharing of the edge resources, while meeting the latency KPI requirements. In this context, data-driven solutions have been implemented at both the VS and the SO, thanks to the provisioning of ML models properly trained within the 5Gr-AIMLP and delivered to the requesting 5Gr-entity.

At the VS layer, instances of services that can share subslices or single VNF instances should do so, in order to reduce the number of active VMs and save both computational and energy resources. However, only services with similar latency requirements should share VNFs, since the more stringent delay constraint is maintained also for the least demanding service, which implies that some of the processing allocated to handle this latter request is not minimal. Such additional capacity can be seen as derived from latency dissimilarity and it is removed when all services sharing a VM have the exact same latency constraint. Solving the sharing or not sharing dilemma is a difficult task, as the optimal sharing policy also depends on the system load. The higher the number of service instances that could share a VM and that need to be deployed, the finer the distinction among latency requirements should be. Instead, the lower the service load, the less exclusive the system should be. This issue can be effectively addressed through a slice sharing algorithm within the Arbitrator (see Section 3.2.1.5) leveraging a latency-based classification of the service instances to be deployed, and dynamically adjusting such latency classification to the system load through an ML-driven approach.

3.4.3 VS layer federation

The 5Gr-VS component of the 5Growth MANO stack has been extended to support VS layer federation (across multiple domains) at vertical (sub)service and network slice levels, as detailed in Section 3.1.6. This innovation is the basis for the 5G-EVE and 5G-VINNI integration and will therefore be used in the pilots using the ICT-17 experimentation platforms.

For the pilots leveraging 5G-EVE, namely INNOVALIA and COMAU, the end-to-end service is divided in vertical (sub)services. One of this (sub)services is designed as one 5G-EVE experiment and onboarded into the 5G-EVE platform through the 5G-EVE Portal GUI. The end-to-end vertical service is onboarded and customized using the 5Gr-VS. During the service end-to-end service instantiation, the 5Gr-VS determines the specific 5G-EVE experiment customization, and requests the experiment allocation and instantiation.

Additionally, the multi-domain interactions considerations addressed within the scope of the project were taken one step further by assessing the potential of placing verticals in a more prominent and direct position in regard to provisioning full E2E orchestration in multi-domain environments. The Transportation and Energy pilots are examples of where an inter-domain network service deployment capability would be beneficial, as they involve the need to provide connectivity between geographically widespread physical assets and services that can be within the coverage span of different mobile operators Therefore, this innovation would allow the creation and deployment of the network service stretching over different operators, in a single procedure, and acted by the vertical (with the necessary API support from the operators).

3.4.4 VS dynamic service composition

The MANO stack developed in the 5Growth project enables a flexible mapping between the vertical services and the constituent vertical (sub)services, network slices, and network slice instances. This

flexible mapping is key for the pilots relying on multidomain splits at the communication service or at the network slice level.

More specifically, the VS dynamic service composition was developed in the 5Growth project and can serve all the Efacec pilots. In the first release of the 5Growth platform, the integration between 5Gr-VS(CSMF) and 5G-VINNI(NSMF) has been implemented, which allows the SONATA Platform to instantiate, terminate or query for previously onboarded NSDs. In the second release the objective is to create all NSDs dynamically.In particular, the second release of the 5Gr-VS enables support for the management of the policies used to determine the vertical service translation into network slices and network services. This means that the policies can be updated dynamically, to improve the dimensioning of the network slices and network services assigned to the vertical service (based on the specific vertical service requirements) if required.

3.4.5 SO automatic network service management

The 5Growth project created a complete MANO stack that integrates various innovations towards automated network service management for handling a variety of services. In this sense it could potentially be of interest in any of the project pilots.

More specifically, at the 5Gr-SO orchestration level (e.g., in INNOVALIA and COMAU Pilots), network services are automatically deployed in the spots decided by placement algorithms based on service requirements and available infrastructure resources. It is integrated with the monitoring platform so that those services are continuously monitored to potentially trigger SLA management operations (e.g., scaling) either based on rules or through AI/ML models downloaded from the AI/ML platform, which is also integrated with the 5Gr-SO. In this way, a closed-loop adaptation is carried out, which links this framework with the self-adaptation actions described in the following section. Furthermore, the metrics based on which automated decisions can be made can also be forecasted through the integration of the 5Gr-SO and 5Gr-VoMS with the forecasting building block.

In 5G-VINNI, since it uses the SONATA Service Platform, the life cycle of each NS/VNFD is natively supported, so it is possible to use its monitoring module to collect metrics from VMs/Containers, compare with pre-defined alarms and trigger actions, like scaling out or in. This process is fully automated and can be applied to the Efacec pilots.

3.4.6 SO self-adaptation actions

The same framework explained for the 5Gr-SO in the previous section is exploited for taking self-adaptation actions to comply with the SLA requirements of the deployed services. In fact, the component of innovation I4 related to data-based automated network service management and self-adaptation actions can be applied to all defined pilots and their use cases. In such industrial environments, with very tight and specific service reliability constraints, it is essential to provide orchestration architectures enabling the automated configuration and collection of data and status from deployed virtualised network services and the underlying infrastructure (computing, storage and networking resources) supporting them. This information can be exploited to perform real-time

data analytics with the help of AI/ML-based models and trigger corrective actions according to agreed SLA requirements, hence ensuring the correct working of the deployed service. An example of the application of 5Growth architecture in such context can be found in [81].

For instance, for those pilots with an interaction with 5G-VINNI, the same process as in the previous section, applied with resource metrics, can be applied to the Efacec Pilots. But, as mentioned, this innovations is transversal and of potential interest for all pilots.

3.4.7 SO dynamic monitoring orchestration

From a general point of view, the features implemented in the 5Gr-VoMS for the collection of metrics and logs allows the full control and lifecycle of the monitoring processes deployed in the vertical services, then enabling the dynamic monitoring orchestration of these processes.

This feature is available in 5G-VINNI through the use of the SONATA Service Platform by using the Element Management System (EMS) for each VNF. This EMS may install all the needed probes and configure the metrics when VNFs start operating. These probes/metrics can be specific for each VNF if that information is available in the VNFDs/NSDs.

For the monitoring of infrastructure metrics in the Innovalia use case, this feature is provided thanks to the capabilities offered by the 5G-EVE Monitoring Platform, being able to collect these infrastructure metrics by using specific network probes, publishing data to the Broker System used in 5G-EVE, based on Apache Kafka, to deliver the monitoring data to the interested entities. In this case, 5Gr-VoMS would consume that data after applying a specific translation of the 5G-EVE data model to fit in the 5Gr-VoMS data model. This process will be explained with more detail in D4.3 for the case of the Innovalia use case.

3.4.8 SO geo-location dependent federation

Verticals often request a service to be deployed with a service coverage for a specific geographical area. The 5Gr-SO based on the abstracted view from the 5Gr-RL is able to realize the service deployment in the local.

Additionally, if there are not available resources in the local domain, the 5Gr-SO can request a federation , for instance, to cover other geographic areas. Furthermore, as part of the 5Gr-SO-to-5Gr-RL interaction, the location of the PoPs is passed to the 5Gr-SO so that it could be exploited for orchestration-related decision making. This allows 5Gr-SO to select and allocate the resources (via 5Gr-RL) on specific VIM/Radio domains.

3.4.9 RL PNF integration

Integration of PNF in 5Growth stack allows supporting all network functions that run on physical infrastructure devices composed by ASIC/FPGA. Typically, in such devices, there is a piece of software (firmware) that is already installed and running.

The feature is studied for radio physical devices (like antenna, baseband units) that allows any pilots to tune the firmware by applying specific configurations (like adding routing table rules for handling PNF traffic).

Each pilot can use it if it needs to interact with physical devices.

3.4.10 RL geo-specific resources

The 5Growth stack handles resources that have to be allocated on a specific NFVI PoP. Since the 5Gr-RL handles multiple domains and multiple PoPs, it allows managing resources that have geographical location constraints. An example is the "Coverage Area" parameter for RAN network slice. Such parameter is optional, but when used, it defines the area where the service is located. For such scope, the 5Gr-RL should provide resources on NFVI PoPs that covers such area (for example uses an antenna whose cell covers the area represented in the "Coverage Area"). Using this feature, all pilots can request 5Growth stack to allocate resources near a specific geographical location.

3.4.11 RAN support

On the one hand, radio network functions (both VNF and PNF) need to load additional configuration after they are up and running. Such configuration element is called Management Function or MF and make the network function ready to handle traffic of the network slices. 5Growth supports the handling of MFs and all pilots can use them when configuring radio network functions.

On the other hand, open RAN, enabled by the support of O-RAN building blocks and interfaces, provides advantages to all pilot use cases because it allows higher competition in the RAN infrastructure, which in turn enables lower prices and higher customization. Moreover, RAN virtualization provides higher flexibility and finer-grained resource allocation to network slices, which can be used in most pilot use cases of 5Growth to enhance the RAN performance.

3.4.12 Integral security

This innovation is able to serve as the underlying security mechanism between the Vertical Slicer and other key network operations elements, such as the orchestrator of other domains containing the network services and connectivity necessary to support the vertical services. This innovation provides the added capability of ensuring the network control interaction between remote peers, thus paving the way for a new kind of safety-certified classification, which is appreciated in the public utility sector. The I11 security innovation will be deployed in the Transportation and Energy pilots, where the SONATA driver belonging to the 5G-VINNI deployment will be used as the deployment domain.

3.4.13 Continuous development and integration

The 5Growth CI/CD innovation is a customizable framework, allowing, among other things, automatic and manually triggered 5Growth platform deployment. It provides a set of artifacts, able to deploy the 5Growth platform in different environments. Those artifacts provide simplified, automated, and repeatable deployment regardless of whether it is the whole 5Growth platform or some of its components, and as such, it is of general application. In this way, WP3 can use this platform to provide appropriate customization of the baseline 5Growth platform for the specific needs of the pilot use cases.

4 Performance Evaluation and Validation

To go one step further on quantifying the enhancements brought by the aforementioned 5Growth innovations, this section presents a great variety of performance evaluation and validation results based on proof-of-concept scenarios involving the innovations introduced in Section 3. Specifically, each performance evaluation topic will include not only the experimental outcomes but also the corresponding setups/approaches applied in its considered scenario(s) to obtain the resulting measurements. Moreover, since some topics cover more than one 5Growth innovation, Table 12 provides the cross-reference among the performance evaluation topic, the corresponding 5Growth innovation(s), the related subsections in Section 3 and Section 4, and the main result of each topic.

TABLE 12: MAPPING BETWEEN PERFORMANCE EVALUATION TOPICS AND RELATED SECTIONS

5Growth Innovations	Performance Evaluation Topic	Subsection in Section 4	Mapped subsections in Section 3	Main result
I1	Automatic Orchestration of End-to-end RAN Network Slicing	4.1	3.1.1	The E2E orchestration time takes less than 30 seconds, in which the 5Growth components only take less than 3 seconds.
I2	Vertical Service Monitoring	4.2.1	3.1.2	Custom Prometheus push gateway brings 38.89% saving in terms of the required storage space in comparison with the native one.
I2+I3	Log Monitoring Pipeline workflow in the 5Gr-VoMS platform	4.2.2	3.1.2 & 3.1.3	The proposed workflow for the Log Monitoring Pipeline is validated, and thus the management of log monitoring in the 5Gr-VoMS is ready for further experiment.
I4+I5	Closed-Loop Automated Service Scaling	4.3	3.1.4 & 3.1.5	The closed-loop workflow among 5G-AIMLP, 5Gr-SO, and 5Gr-VoMS for the automated network service scaling is validated. AI/ML-related operations take extra 3.5% and 10% time during installation and scaling operation, respectively.
I6	5Growth-5G-EVE Service Deployment Evaluation	4.4	3.1.6.1.3	The instantiation and termination operations across domains can be completed in about 3.10 minutes and 2.55 minutes, respectively.
	Interdomain Service Deployment	4.5	3.1.6.1.3	The performance of inter-domain vertical slice deployment is decided by the slowest domain, and it takes on average 8.16 minutes in the two considered domains.

	DLT Federation in 5Growth	4.6	3.1.6.1.3	The federation between multiple ADs can be managed by a single smart contract, and it takes around 90 seconds to resolve 30 federation announcements with a total of 50 concurrent bidders.
I4+I6	SLA management via Scaling Requests of Composite Network Services implying NSF	4.7	3.1.4 & 3.1.6.1.2	The scaling operations (scale in and scale out) in multi-AD deployment scenarios take less than 40s, which is in line with the expected KPIs.
I7	Next-generation Radio Access Networks	4.8	3.1.7	The results show up to 20% OPEX savings in terms of computing resource utilization, and 30% CAPEX savings in terms of computing infrastructure reduction.
I8	GA-based SFC placement	4.9.1.1	3.2.1.1.1	The GA-based solution can almost reach the optimal E2E service delay. After applying the location-aware method, E2E delay is decreased by 30% with much less execution time than the location-agnostic one.
	VNF Autoscaling	4.9.1.2	3.2.1.1.2	The MLP-based solution can reach 88.98% and 90.38% accuracy, respectively for fast browsing and video streaming services.
	AI-Driven scaling on of C-V2N Services	4.9.1.3	3.2.1.1.3	The LSTM approach achieves the highest reward since it can use the lowest average number of CPUs without under-dimensioning for the incoming workload.
	Slice Sharing at 5Gr-VS Arbitrator	4.9.1.4	3.2.1.1.4	Dynamic slice sharing at the 5Gr-VS brings computing resource saving, particularly a saving of 30% is made in low load conditions.
	5Gr-SO - 5Gr-RL: Resource Abstraction and Allocation Algorithms	4.9.2.1	3.2.1.2.1	Proposed CSA and InA operational modes at the 5Gr-RL lead to different trade-offs between abstraction accuracy exposed to the 5Gr-SO and efficient network resources usage in the WAN.
	Performance Isolation for Network Slicing	4.9.2.2	3.2.1.2.2	Slices performance isolation for transport network guarantees both bandwidth and delay requirements over a mixed data path comprising OvSs and P4 switches.

	Dynamic Profiling Mechanism	4.9.3	3.1.7	The effectiveness of DPM is shown to have 71.25% overall profile forecasting accuracy considering 3 different data entities (Device, Service, Network).
I9	Anomaly Detection	4.10	3.2.2	The anomaly detection framework achieves an average 87.51% prediction performance from different granularity levels (in terms of distinct anomalous behaviours).
I10	Forecasting	4.11	3.2.3	AI/ML-based forecasting schemes show improved accuracy than the time series analysis ones when the time series present large variations in the data pattern, and they can be used by the 5Gr-FFB module.
I11	Moving Target Defense	4.12	3.3.1	MTD mechanism provides safer access to the fundamental services, allowing the interaction between 5Growth's capabilities with other players, e.g., between verticals and network service provider directly.
I12	5Growth Platform Deployment CI/CD	4.13	3.3.2	The new containerized 5Growth platform deployment CI/CD pipeline is being performed 5 times faster compared to the VM-based procedure.

4.1 Automatic Orchestration of End-to-end RAN Network Slicing

In this section, we evaluate the implementation of Innovation I1 described in Section 3.1.1 in terms of processing time of each layer in the 5Growth architecture. In the following, we first describe the measurement environment and methodology and then show the result.

4.1.1 Measurement Environment and Methodology

5Growth stack runs as Docker containers on a Linux machine. Cloudify v5.1.0 is used as open-source core MANO to support the NFV orchestration of 5Gr-SO. The test environment uses stub domain for radio, transport, and cloud. These domains include the processing of resource management commands, without considering the execution time of commands specific of network controller (that are not modified by the RAN slicing implementation).

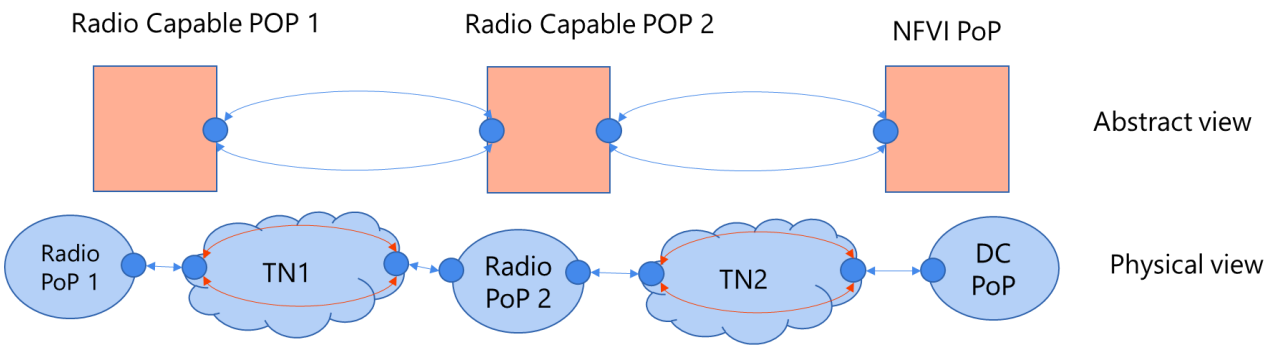


FIGURE 52: PHYSICAL AND ABSTRACT TOPOLOGY USED FOR MEASUREMENT

Figure 52 shows the physical topology created with the stub domains that consists of two radio domains (one used for running RAN function and the other used to run CN function) and a cloud domain (to run vertical applications). Such domains are connected via transport network domain; each transport domain uses a ring topology.

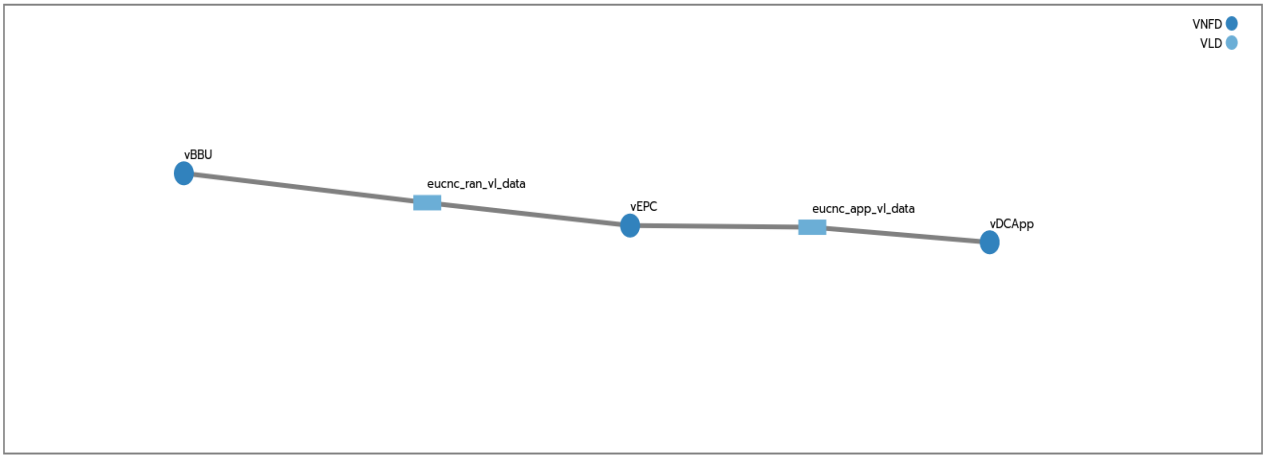


FIGURE 53: NSD USED FOR MEASUREMENT

The NSD of the service used for measurement is shown in Figure 53. The service follows the 3GPP Network Slice (NS) model and includes a RAN VNF (called virtual Base-Band Unit or vBBU), a Core VNF (virtual Evolved Packet Core or vEPC) and a generic vertical application (vDCApp).

Each module of 5Growth stack (5Gr-SO, 5Gr-VS and 5Gr-RL) uses a logging system that reports in a configuration file the output of processed commands (e.g., NS instantiation and termination). The processed command also reports the system time when the specific command is received, processed and execution is finished. As the time is reported for each internal method invoked in NS command processing, this method of measuring allows to perform a profile of the 5Growth stack modules.

4.1.2 Measurement Results and Evaluation

The measure is obtained repeating the NS instantiation and termination procedure 20 times. Table 13 reports the results with the maximum value obtained.

TABLE 13: PROCESSING TIME MEASUREMENT RESULTS

5Growth Module	Processing time	
	NFV-NS instantiation	NFV-NS termination
5Gr-VS	max: 975ms	max: 955ms
5Gr-SO	max: 980ms	max: 930ms
5Gr-RL	max: 995ms	min: 990ms
Cloudify	max: 20s and 175ms	max: 20s and 175ms

The results show that the impact, in terms of the extra processing time introduced by 5Growth modules is small and negligible when comparing with the orchestration time produced by the open-source core MANO (that is 20 times higher). That is also true if the domain controller processing time is considered (for example the instantiation of a VM in a datacentre can take up to several minutes depending on how many internal processes are started). It is also worth noting that about 80% of the time budget is used for the communication with the external MySQL database to store and the NS info and records the history of commands. This means that the functions extended to implement the innovation are high performing.

4.2 Deployment and Orchestration of the Monitoring Platform

4.2.1 Vertical Service Monitoring

The custom Prometheus push gateway that was developed and described in Section 2.5 gives the possibility to use Prometheus metric storage space more effectively. The effectiveness of a custom Prometheus push gateway usage, comparing with native Open-Source Prometheus push gateway is described with the following use-case example.

The example use case considers that the 5Gr-VoMS has 3000 abstract sources of metrics to monitor from many abstract VNFs (for this purpose, the metrics type or the number and type of VNFs do not matter). They are split into three equal groups. The first group of metrics generates metrics every 5 seconds, the second group every 10 seconds, and the third group every 15 seconds. The metrics are collected by both the Custom Push Gateway and the Native Push Gateway to compare the results. The huge downside of the Native Open-Source Prometheus Push Gateway is that it supports only one scrape interval for all metrics. As such, all the metrics must be gathered with the smallest interval to avoid losing data. This scrape interval is enough for the first group, but it is redundant for the second and the third groups. Metrics of the second and third groups hence will use more storage space to hold metrics than really needed. This connection between scrape interval and metrics changes is shown in Figure 54.

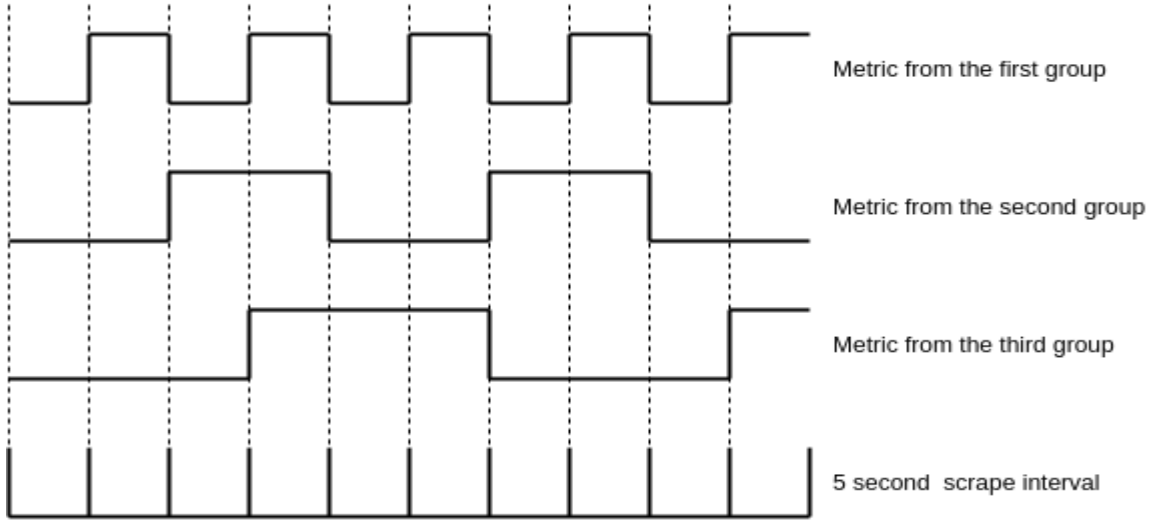


FIGURE 54: THE CONNECTION BETWEEN SCRAPE INTERVAL AND METRICS CHANGES FOR NATIVE OPEN-SOURCE PROMETHEUS PUSH GATEWAY

The custom Prometheus push gateway can work with metrics of different scrape intervals. Prometheus stores an average of only 1-2 bytes per sample (average sample size is based on average metrics payload observed on different Pilots and Demos run during the 5Growth project). As for the assumption, 2 bytes per sample will be used for storage space calculation. Thus, to plan the capacity of a Prometheus server, the following formula can be used:

*needed disk space = ingested samples per second * bytes per sample*

The calculated required storage space for two types of Prometheus push gateway is shown in both Table 14 and Figure 55. By comparing the amount of required storage space for the native Prometheus Push Gateway and the custom Prometheus Push Gateway, we can note that this 38.89% gain, as the effectiveness of custom Push gateway column shows in Table 14.

TABLE 14: REQUIRED STORAGE FOR TWO TYPES OF PROMETHEUS PUSH GATEWAY

Time (Days)	Required storage space (Megabytes)		Effectiveness of custom Push gateway (Percentages)
	Native Prometheus Push gateway	Custom Prometheus Push gateway	
0	0	0	0
5	494.38	302.12	38.89
10	988.77	604.25	38.89
15	1483.15	906.37	38.89
20	1977.54	1208.50	38.89
25	2471.92	1510.62	38.89
30	2966.31	1812.74	38.89

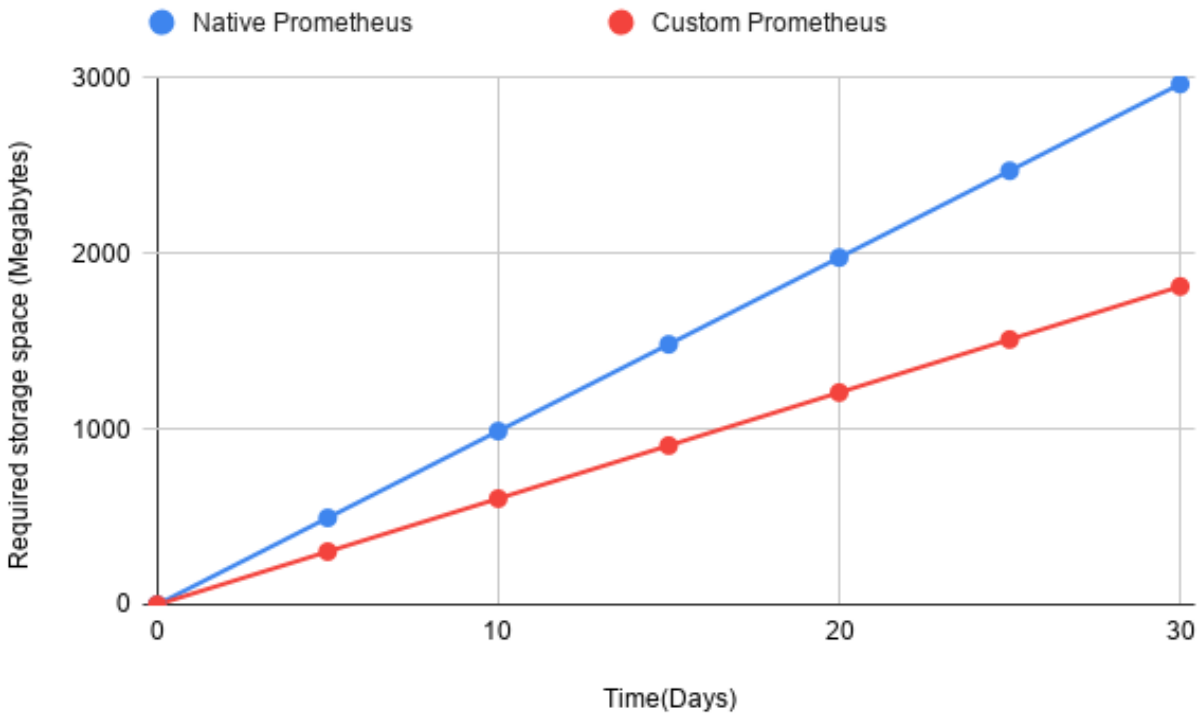


FIGURE 55: REQUIRED SPACE FOR TWO TYPES OF PROMETHEUS PUSH GATEWAY DEPENDING ON STORAGE TIME

4.2.2 Log Monitoring Pipeline workflow in the 5Gr-VoMS platform

In order to verify whether the log monitoring pipeline has been correctly configured for its subsequent integration into the 5Gr-VoMS platform, a proof of concept emulating the requests sent by the Config Manager to the Log Pipeline Manager is examined in the following paragraphs. The overview of the software components used for such examination, whose complete architecture and workflow is explained in D2.4 [9], can be seen in Figure 56.

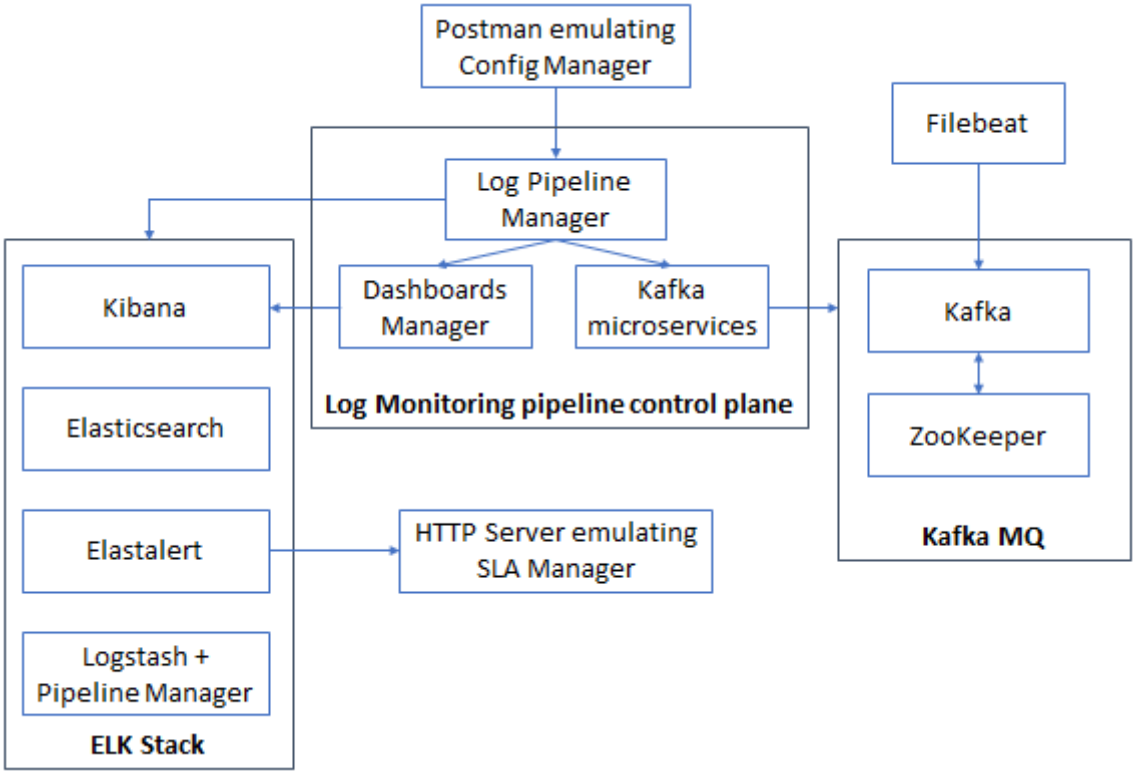


FIGURE 56: COMPONENT OVERVIEW FOR VALIDATING THE LOG MONITORING PIPELINE

In particular, all the software modules, excepting Filebeat [74] and Postman [75] emulating the Config Manager, have been deployed as containers orchestrated by Kubernetes, following some guidelines to setup the scenario and run the different scripts to test the endpoints exposed by the Log Pipeline Manager towards the Config Manager. These guidelines are also explained in D2.4 [9], as they are available in the corresponding software repository containing the 5Gr-VoMS.

A set of Postman collections is used to emulate the messages sent by the Config Manager to the Log Pipeline Manager. Regarding the workflow to configure the monitoring of a given log in a specific VNF, Filebeat plays this role, being directly installed and configured in the server hosting the Kubernetes to directly interact with Kafka through particular port, exposed by Kubernetes for this proof-of-concept.

From a high-level point of view, the applied steps in this demo are the following:

1. Deploy all Kubernetes to create the corresponding containers. All these deployments expose names that are directly resolved by the Kubernetes built-in DNS, so that the different containers can make use of these names to directly refer to a specific container of the scenario, without the extra IP address handling.
2. Configure Kafka container and link it to the ZooKeeper instance running in the scenario.
3. Configure all components within the ELK stack (cf. Figure 56), together with the Dashboards Manager. Such Dashboards Manager entity is in charge of interacting with Kibana to create the dashboards that provides the collected logs in a fashion view.
4. Configure Kafka microservices, together with the Log Pipeline Manager.

5. Create a new log monitoring job related to a given network service, i.e., send a Postman request to the Log Pipeline Manager. This will trigger the creation of the Kafka topic, the Logstash pipeline or the Elasticsearch index, among others.
6. Create a Kibana dashboard for that network service and then create the objects in Kibana through the Dashboards Manager to properly display the monitoring data after reception.
7. Create an Elastalert rule that checking if one specific message collected by the ELK Stack contains the number "3" in a per-20 second interval.
8. Create a Log Scraper to extract data from the topic created, publishing the messages that contains the number "4" in a new Kafka topic created off-line with the corresponding microservice.
9. Configure Filebeat to start monitoring a specific file that will be fed in the next steps. After this step, Filebeat is started to run.
10. Generate the data that includes a list of numbers, from 1 to 10, in separate lines in the file monitored by Filebeat. Such data is then sent to Kafka, being collected by Logstash and saved by Elasticsearch. As a result, two main operations are triggered consequently:
    - The dashboard starts presenting data, as it can be seen in Figure 57. In that case, it presents the same information but using three kind of plots: (1) A pie chart on the left-hand side presenting the percentage of logs received for each VNF related to a given network service. In this example, there are two VNFs (Filebeat was used in two different servers publishing to the same Kafka container), and in one of them, there are two different monitored logs (via using Filebeat to monitor several log files), (2) A horizontal bar diagram on the right-hand side presenting the total number of messages received for each monitored log (for each VNF), and (3) A search dashboard presenting the content of each message received in the platform.
    - As the numbers from 1 to 10 are sent from Filebeat, it meets the configured Elastalert rule. Therefore, the action configured is to send a message to a dummy HTTP server, emulating the SLA manager service from the 5Gr-SO, which only displays that a message has been received with a specific timestamp. A sample output would look like:

```
04/05/2021 14:27:52 INFO Request received - POST /alert_receiver
04/05/2021 14:27:52 INFO Data received: {'startsAt': 'Tue May  4
14:27:52 UTC 2021', 'alertname': 'c
023539a-ace4-11eb-b80c-e2b576bed96e'}
```

    - Finally, by running a Kafka Consumer in the topic created for testing the Log Scraper capabilities, it was checked that the messages generated by the Log Scraper were sent correctly to Kafka. A sample output would look like:

```
{"record":[{"agent":"be9d9608-db3b-4fe0-b9c0-
0554f59adc30","log_path":"\/var\/log\/test2.log","host":"5growt
h-k3s-master-monit","message":"4","timestamp":"2021-05-
04T14:27:50.260Z"}]}
```

10. Delete the log scraper, the alert rule, the dashboard and the job created (following that order), releasing all the resources.

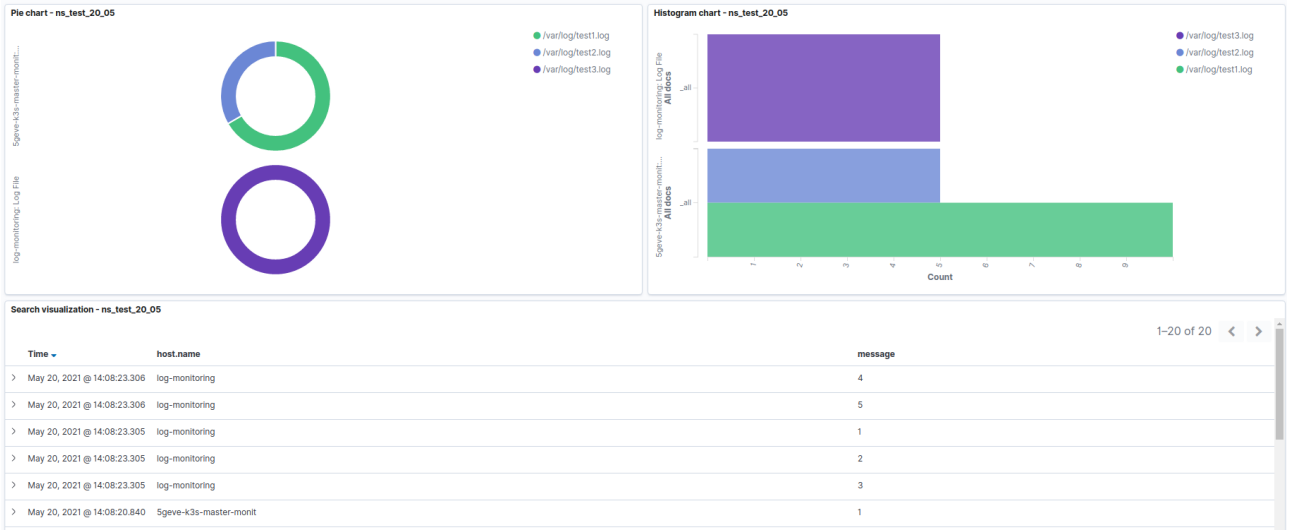


FIGURE 57: EXAMPLE OF A KIBANA DASHBOARD FOR A GIVEN SET OF MONITORED LOGS

To conclude, the above proof-of-concept verifies the correctness of the Log Pipeline Manager in terms of the interaction with the Config Manager and other components, which is set as the key step toward the final integration within the 5Gr-VoMS.

4.3 Closed-Loop Automated Service Scaling

This section presents the PoC validation of the AI/ML-based closed-loop scaling operation introduced in Section 3.1.4, and the full video can be found in [82]. For this validation [83] we use a complete implementation of the 5Growth platform where we deploy a Digital Twin (DT) NFV-NS, as relevant representative of Industry 4.0 services [84]. As shown in Figure 58, the DT application is composed of two VNFs. VNF1 creates a rendering of the connected robots based on the geographical coordinates received from the robots themselves, allowing a human user to control the robots' movements through a joystick. VNF2 translates the commands inserted by the human user into instructions for the robots. Between the two, VNF2 is the most critical, as its processing time must be below 60ms to ensure a timely delivery of the movement instructions to the robots.

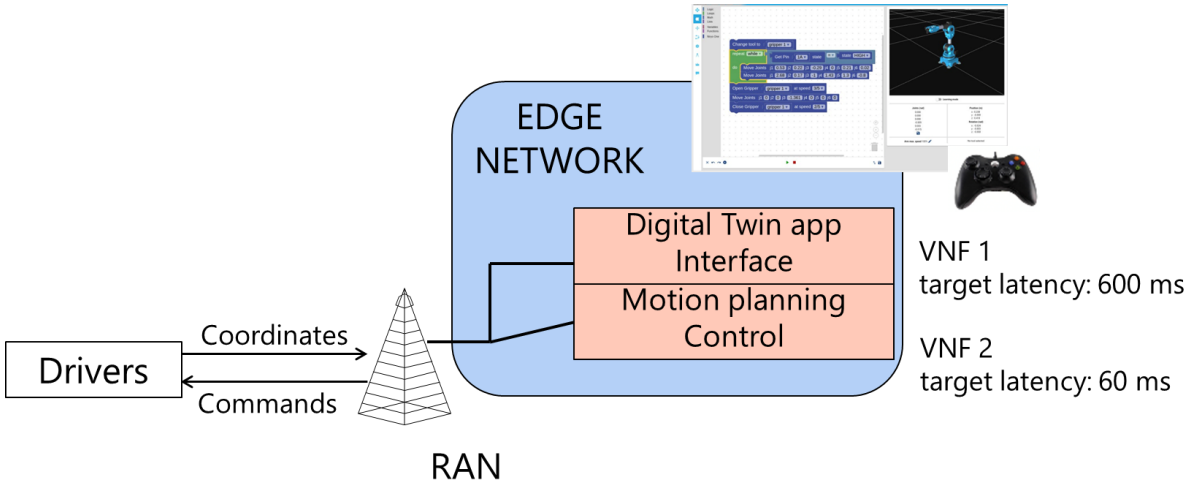


FIGURE 58: DIGITAL TWIN APPLICATION USE CASE

To meet such target latency, the NFV-NS presents multiple Instantiation levels (ILs), each corresponding to a different number of VNF2 instances. At deployment-time, the DT NFV-NS consists of one instance per VNF, i.e., IL=1. However, as the number of robots to control increases, so does the CPU load and the processing latency of VNF2, the NFV-NS IL must be properly scaled out to balance the CPU load over multiple instances of VNF2 and keep its processing time below the target values. Likewise, as the CPU load decreases, the NFV-NS IL should be scaled in to save computational resources. A real-time decision on the IL to adopt is therefore required, to trigger a scaling (out/in) operation as needed.

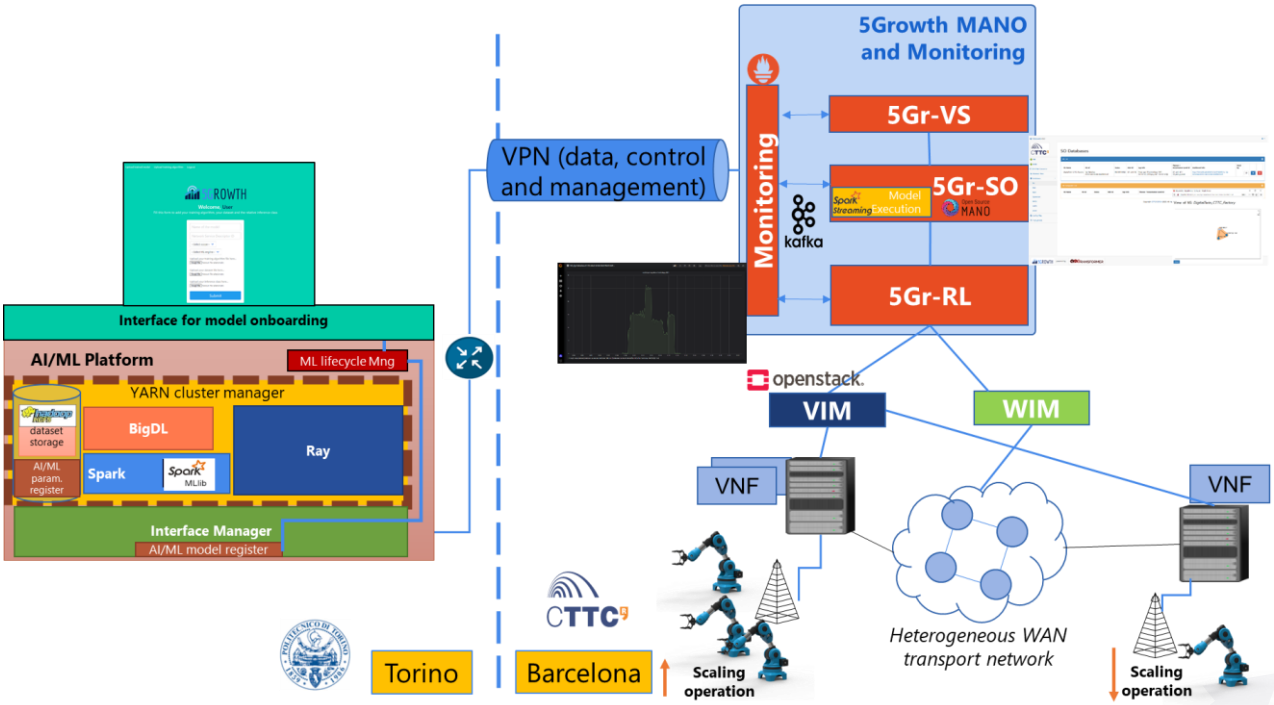


FIGURE 59: EXPERIMENTAL PROOF-OF-CONCEPT

Figure 59 shows the experimental proof-of-concept for the evaluation of the closed-loop automated scaling operation. The PoC has been implemented using an instance of the 5Growth platform (which includes the 5Growth MANO stack, the monitoring platform and the infrastructure) composed of the 5Gr-Vertical Slicer (5Gr-VS), the 5Gr-Service Orchestrator (5Gr-SO), the 5Gr-Resource Layer (5Gr-RL) and the 5Gr-Vertical oriented Monitoring Service (5Gr-VoMS). This instance of the 5Growth platform is completed with: (i) an instance of OSM Release 6 and an instance of Apache Spark (version 2.4.0) paired with the 5Gr-SO, and (ii) an OpenStack-based VIM to manage the VNFs and an ABNO-based WIM controlling a multi-technology (wireless, optical) transport network under the control of the 5Gr-RL.

The 5Gr-AIMLP, serving as AI/ML as a Service (AIMLaaS) platform, is integrated into the 5Growth system to follow the workflow described in Section 3.1.4.2 with the aim of performing a close-loop scaling operation fulfilling the SLA requirements of the deployed DT NFV-NS under evaluation. In this PoC, as mentioned before, the 5Gr-SO performs actions based on the suitable IL decided by the inference job running in Apache Spark according to the measured service demands of the deployed DT NFV-NS instance. To do so, the 5Gr-SO uses an AI/ML model (a random forest algorithm) that is downloaded from the 5Gr-AIMLP during NFV-NS instantiation and that is fed with the real-time CPU load values provided by the 5Gr-VoMS all the way through the data pipeline. The AI/ML model is trained offline within the 5Gr-AIMLP using Apache MLlib and a training dataset with 318K entries (total size: 20.2 MB). The resulting model is packed in a zip archive of 86.5KB (46.4 KB are devoted to the auxiliary file needed for inference).

TABLE 15: 5GR-AIMLP OPERATION AND INTERACTION WITH 5GR-SO

5Growth Building Block	Operation	Measured Time
5Gr-AIMLP	Dataset upload (offline)	10.3 ± 2.0 s
	Training (offline)	35.6 ± 2.5 s
5Gr-SO	Model/Aux file download (instantiation phase)	281.9 ± 20.4 ms
	AI/ML-related operation (instantiation phase)	1367 ± 40.9 ms

Table 15 presents the measured times (average and standard deviation) over 10 repetitions of the experiments. In general, the time required by the 5Gr-AIMLP to upload the dataset and train the model (offline process) depends on the file size and the server used; under the above settings and using an Intel i7-4790 CPU with a 32 GB-RAM, it is in total around 46±4.52s. During NFV-NS instantiation, the 5Gr-SO requires 281.9±20.38ms to download the files from the 5Gr-AIMLP (online process), which represents a total of 20.6% of the average time devoted by the 5Gr-SO to set up the data engineering pipeline for AI/ML-based scaling operations [85] which in this evaluation is 1367±40.9ms. However, configuring the pipeline is just a small part of the instantiation process, with VNF deployment being instead the most time-consuming step [86]. In fact, the impact of AI/ML-

related operations on the whole instantiation process is limited to about 3.5% of the total average instantiation time measured at the 5Gr-SO (39.78±3.24s).

Once the AI/ML model is running, the SLA manager compares the current IL of the service with that inferred by the AI/ML model (i.e., the one the NFV-NS should be running to deal with a given CPU load) and, in case there is a difference, a scaling operation is triggered (e.g., scale out/in if more/less resources are needed). When this happens, the scaling operation starts following the workflow as described in Section 3.1.4.2 and in [85], then the system takes care automatically of stopping/starting/reconfiguring the data engineering pipeline and monitoring jobs as needed. Note that during scaling, there is no need for interaction with the 5Gr-AIMLP.

TABLE 16: AI/ML-BASED SCALING OPERATION IN 5GROWTH

Operation	Measured Time	AI/ML impact
Scaling out	26.72 ± 2.75 s	9.55%
Scaling in	24.54 ± 2.58 s	10.32%

Next, we focus on the impact of AI/ML operations during subsequent scaling operations at the 5Gr-SO. Table 16 presents the average time and standard deviation over 10 repetitions of the scaling out/in operation, i.e., moving from one to two instances of VNF2 and from two to one instance, respectively, and the percentage of time devoted to AI/ML operations in the scaling workflow. The impact of AI/ML operations is now around 10%, i.e., much more than in the case of instantiation. This is mainly due to the time it takes to stop the inference process while a new VNF instance is created/removed and the rest of the NFV-NS (connections and monitoring jobs) is updated accordingly. Stopping such process takes around 2.49±0.17s and it is performed to avoid overruling of a decision on the IL in the middle of a scaling workflow execution (i.e., in a transient period).

4.4 5Growth-5G-EVE Service Deployment Evaluation

This section reports on the experimental results of the proof-of-concepts comprising the integration of 5Growth and 5G-EVE platforms. This proof-of-concept allows a preliminary evaluation of two of the 5Growth inter-domain interactions, namely the *communication service level* and the *network service level* interactions. At the communication service level, the 5Growth and 5G-EVE embody the roles of consumer Communication Service Management Function (CSMF) and the provider CSMF, respectively. At the network service level, 5Growth and 5G-EVE embody the provider service orchestrator (SO) and consumer SO, respectively. The evaluation scenario presented in Figure 60 is implemented in the 5TONIC lab [98], which provides not only a 5G network implementing 5G non-standalone (NSA) access (BB630 baseband and Advanced Antenna System AIR 6488) but also a datacentre that hosts both 5Growth and 5G-EVE platforms as independent domains.

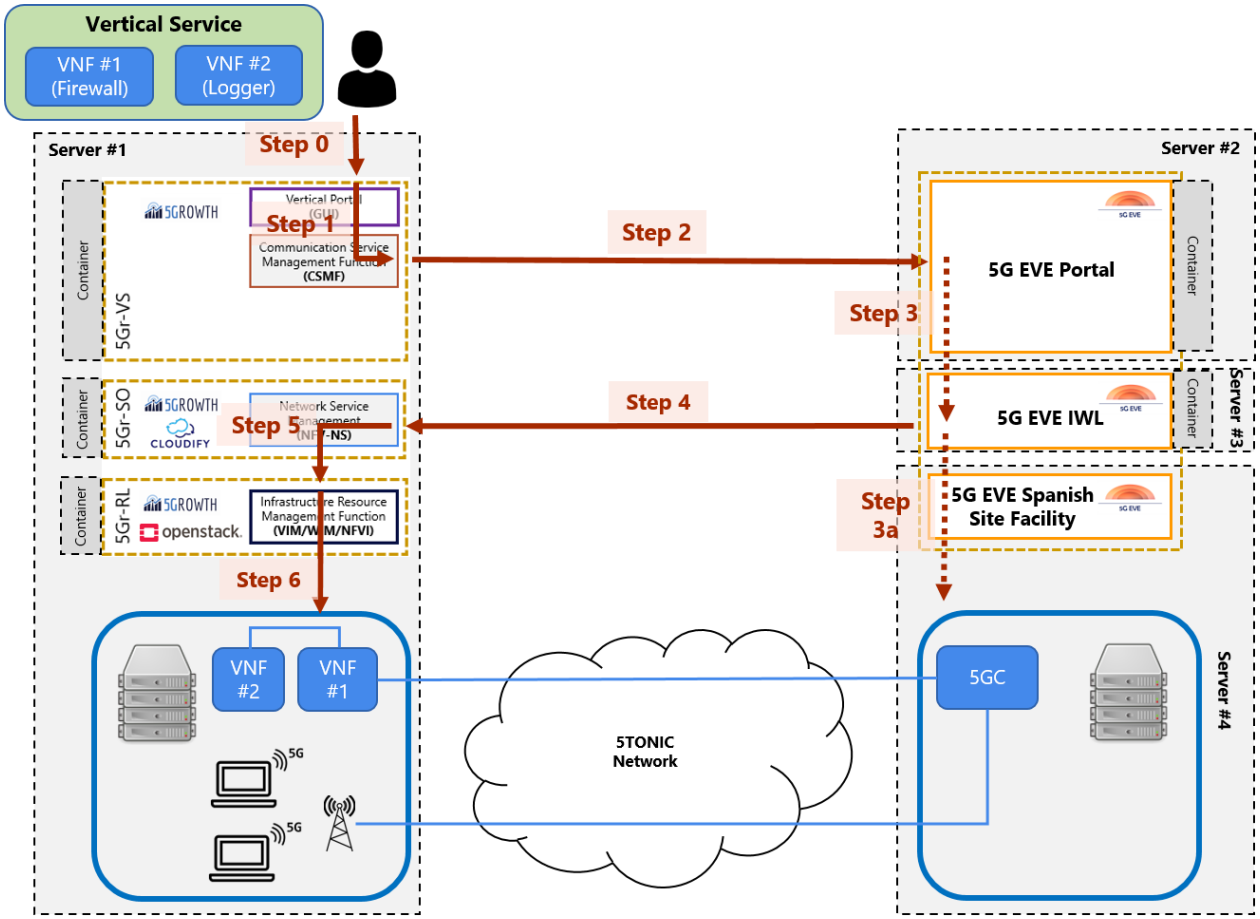


FIGURE 60: 5GROWTH - 5G-EVE EVALUATION SCENARIO

This evaluation scenario aims at providing a baseline reference for both the *communication service level* and the *network service level* inter-domain interactions. As such, it considers a simple vertical service composed of only two VNFs. Nevertheless, more complex vertical services can be devised (e.g., composite and/or multi-site services) that reflect the real needs of the vertical industries. This experiment implements both instantiation and termination lifecycle management operations, which workflow is depicted in Figure 60:

0. **Step 0:** The user requests the vertical service instantiation or termination at the 5Gr-VS Portal. This step sets the initial time for the experiments.
1. **Step 1:** 5Gr-VS decomposes the vertical service into multiple sub-services and delegates their provisioning and management to an external 5G-EVE domain, including translation and mapping of all the required requests.
2. **Step 2:** 5Gr-VS issues vertical service instantiation or termination requests towards the 5G-EVE Portal, as well as monitors their execution status.
3. **Step 3:** 5G-EVE manages the decomposition of the vertical service and issues NFV network service requests towards the 5Gr-SO and/or service orchestrator of other site facilities. This step is omitted from the analysis, as it is 5G-EVE-only step.

4. **Step 4:** 5G-EVE Inter Working Layer (IWL) processes requests related to the interaction with the 5Gr-SO, including translation between ETSI SOL005 and ETSI IFA013 data models and polling (each 10s) the status of operations.
5. **Step 5:** 5Gr-SO processes incoming requests, including creation or elimination of NSs, verification of available resources at the 5Gr-RL, and update of existing databases.
6. **Step 6:** 5Gr-RL (de)allocates the required resources (intra-PoP network) where to deploy the VNFs, including their instantiation or termination.

The time profiling corresponding to each of the previous steps, in terms of both average and standard deviation values, are presented in Table 17. Also in the same table, we can get the total time consumption for both installation and termination operations via the summation from all steps.

TABLE 17: TIME PROFILLING OF 5GROWTH AND 5G-EVE INTEGRATION

		Instantiation operation		Termination operation	
		Average time	Standard deviation time	Average time	Standard deviation time
Communication service level	Step 1	6.171sec	0.803 sec	1.324 sec	0.074 sec
	Step 2	49.344 sec	0.232 sec	86.912 sec	11.075 sec
	Step 3	N/A		N/A	
Network service level	Step 4	5.274 sec	2.687 sec	3.530 sec	3.606 sec
	Step 5	235.4 ms	118.2 ms	84.5 ms	14.8 ms
	Step 6	2.09 min	0.05 min	0.99 min	0.026 min
Total		3.10 min		2.50 min	

From the 5Gr-VS perspective, it will regularly poll in a 60-second period and thus it only detects the vertical service in the execution or termination state approximately 53 s after its instantiation or termination.

Finally, the most time-consuming operation during the vertical service instantiation is related to the creation and instantiation of the VNFs and the creation of the virtual links and networks (step 6), representing about 63% of the total instantiation time. Similarly, the termination and clean-up of the previous aspects (step 6) also account the greater time-consumption, representing about 40% of the total termination time.

4.5 Interdomain Service Deployment

This section illustrates the solution tested making use of the interdomain scenario on a network slice level. This solution implements the network slice multidomain, in which the vertical service deployed to be evaluated is described as a simple service composed of two instances of a probing VNF. This probing VNF will aggregate all sorts of sensorial data (e.g., if theoretically considering the EFACEC transportation pilot, such data could be train and track informational telemetry) in closer proximity with the provider where it is instantiated. The probing VNF makes use of forwarding-plane performance data (throughput and latency) to understand the E2E slice is compliant with the

Verticals' KPIs specification. Instantiating the E2E vertical slice requires an additional VNF to handle a secure tunnel (i.e., a VPN such as Wireguard) that is responsible for the interconnection of the two sub-slices.

In this evaluation, 5Gr-VS instantiates the secure tunnel VNF, an internal component of the 5Growth inter-domain solution, and seamlessly manages it without the vertical's intervention at the network service level. Further enhancements can be made so that the 5Gr-VS can fully support day-1 primitives needed at the NSMF level for the establishment of the tunnel at that level. This is a requirement when the interdomain process is applied into domains featuring the OSM orchestrator in one domain, and a different orchestrator in the other domain. For this evaluation, two domains featuring the OSM orchestrator were considered.

4.5.1 Evaluated mechanism description and evaluation

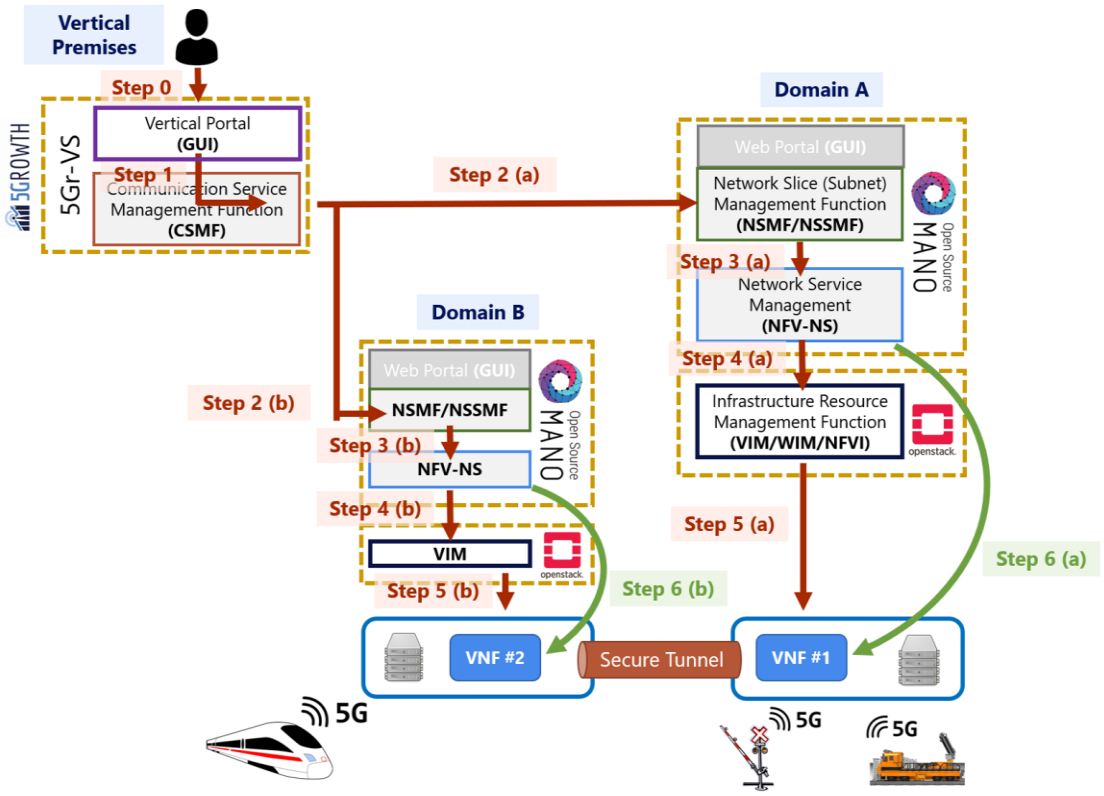


FIGURE 61: EVALUATION OPERATIONAL STEPS

This experiment covers the underlying impactful interactions required to instantiate the inter-domain slice across domains A and B, profiling the various steps' delays. The procedure is represented in Figure 61, and it is divided into the following steps:

0. **Step 0:** The vertical's operator enters the pilot 5Gr-VS, where it needs to select the vertical service to create an instantiate the E2E vertical slice with the two probing VNF instances. He then requests its instantiation. Omitting the operator delays, we can know the time required to show the instantiation page, fully rendered with all the dynamically acquired information

about available templates, resources, and other system states. This page loads on average in 238.32 ± 5.96ms.

1. **Step 1:** The pilot's 5Gr-VS portal will then map the vertical service request into corresponding templates/descriptors and will translate the vertical service request into a request compliant with the 5Gr-VS CSMF. This step considers the elapsed time between pressing the action button in the instantiation form and triggering action in the 5Gr-VS CSMF.
2. **Step 2:** The 5Gr-VS CSMF decomposes the E2E vertical slice into sub-slices. Then it will request their provisioning and management to the different 5Gr-VS NSMF domains, parallelizing the requests to the domains (e.g., A or B in Figure 61). The 5Gr-VS CSMF is also responsible for the instantiation request of the secure tunnel VNF in every sub-slice and instructs the 5Gr-VS NSMF that the tunnel VNF must be configured through the respective primitive with endpoint information that arises from the resolved addresses after VDU instantiation.
3. **Step 3:** The 5Gr-VS NSMF will then forward the sub-slice request into the appropriate artefacts of the domain's NFV-NS (i.e., using network slice templates or composition of network service descriptors). At the same time the 5Gr-VS NSMF will request, in parallel, that each SO instantiates all components required to build the sub-slice. The NFV-NS Life-Cycle Management (LCM) is now delegated to the respective SO. The parallel requests took an average of 4.09 ± 0.70ms in domain A and 3.50 ± 0.52ms in domain B.
4. **Step 4:** The SO determines which VIM is more suitable to fulfil the request based on the descriptors and starts the LCM of the requested components, deploying the service-specific managers and support for service primitives (if the component requires them). In particular, the secure tunnel VNF crucial for the inter-domain connectivity requires a mix of day-1 and day-2 primitives to configure the tunnel endpoints with the resolved VDU information.
5. **Step 5:** The VIM receives the instantiation requests and delivers the virtual resources from its shared resource pool if the resources to fulfil that request are available within the set constraints. Upon instantiation of each VDU, the VIM makes available the resolved endpoint address information. Steps 4 and 5 combined take 62.90 ± 6.74s in domain A and 50.94 ± 8.60s in domain B.
6. **Step 6:** Upon delivery of the virtual resource, the LCM within the NFV-NS will start provisioning the slice/network services accordingly to the vertical requirements and the computed configurations during the network slice decomposition process of the upper layers of the stack. This facility will also enable the on-demand configuration of the vertical's services if those primitives exist. The step takes 421.93 ± 67.26s in domain A and 160.39± 13.73s in domain B.

Table 18 summarizes the obtained results.

TABLE 18: INTERDOMAIN EVALUATION RESULTS PER STEP

	Communication Service level		Network service level		
	Step 1	Step 2	Step 3	Steps 4 & 5	Step 6
Instantiation	9.95± 1.54ms	216.69± 6.67ms	4.09± 0.70ms	62.90 ±6.74s	421.93 ±67.26s

The instantiation of the E2E vertical slice lasted for 8.16 ± 1.07min, with the critical path being determined by the slowest domain (A). The faster domain (B) ends the instantiation phase in 3.61 ± 0.27min and is closer to the same services' instantiation times without the inter-domain connectivity when instantiated on this domain only (3.20 ± 0.37min).

4.6 DLT Federation in 5Growth

In this section, we present a feasibility validation of the DLT federation described in Section 3.1.6.1.3.

To emulate the scenario, we deployed two virtual machines: Domains VM and Blockchain VM. The Domains VM emulates from 2 to 100 administrative domains through a nodeJS/React web server application. On the Blockchain VM, a private Ethereum blockchain is deployed. The Federation smart contract (SC) is developed using Solidity and deployed on the private Ethereum chain, which uses Proof-of-Work (PoW) consensus mechanism and default difficulty increase. Both machines are interconnected in a private Ethernet network. The results are shown at Figure 62.

We applied a lowest offered price policy for the consumer AD to choose the bidder AD as a provider. We executed four sets of experiments:

1. We measured the time it takes for the ADs to register to the Federation SC. The top-left graph shows the results. From the given results, it is clear that the registration time for multiple AD registering at the same time can range from ~ 0.6 seconds up to ~ 3.8 seconds. Even in the extreme case of 100 ADs registering at the same time that is quite uncommon, the setup time takes less than 4 seconds.
2. We measured the federation time for a single federation announcement. The measured time is from the start of the experiment until the selected provider receives the notification to deploy a service. We repeated the experiment for set of [2, 5, 10, 20, 50, 100] bidding ADs, competing to win the reverse auction process, i.e., in the first trial 2 bidding ADs (bidders) compete for the single service, while in the last trial 50 bidders are competing. The results are shown on the top-right graph of Figure 62. It shows that a single service federation can be performed within 5 seconds. Additionally, it describes that the number of bidders have low impact on the time consumption.
3. We repeated the 2nd experiment while creating two federation announcements in sequence. We measured the total time it takes to resolve the two federation announcements while using the set of [2, 5, 10, 20, 50] bidders. The results of this experiment are shown on the bottom-right graph of Figure 62. Starting from 2 bidders per federation announcement the total

completion time linearly increases, with maximum ~ 11 seconds for 50 bidders per each federation announcement.

To justify the results from the previous experiment and stress the system, we measured the total time it takes to resolve set of [2, 5, 10, 20, 30] announcement offers posted. We used constant number of 50 bidders for each trial. These results are shown on the bottom-left of Figure 62. There is a monotonic increase of the total federation time as the number of announced offers increases. The system manages to handle 30 sequentially announced offers with 50 bidders per announcement within total of ~ 90 seconds.

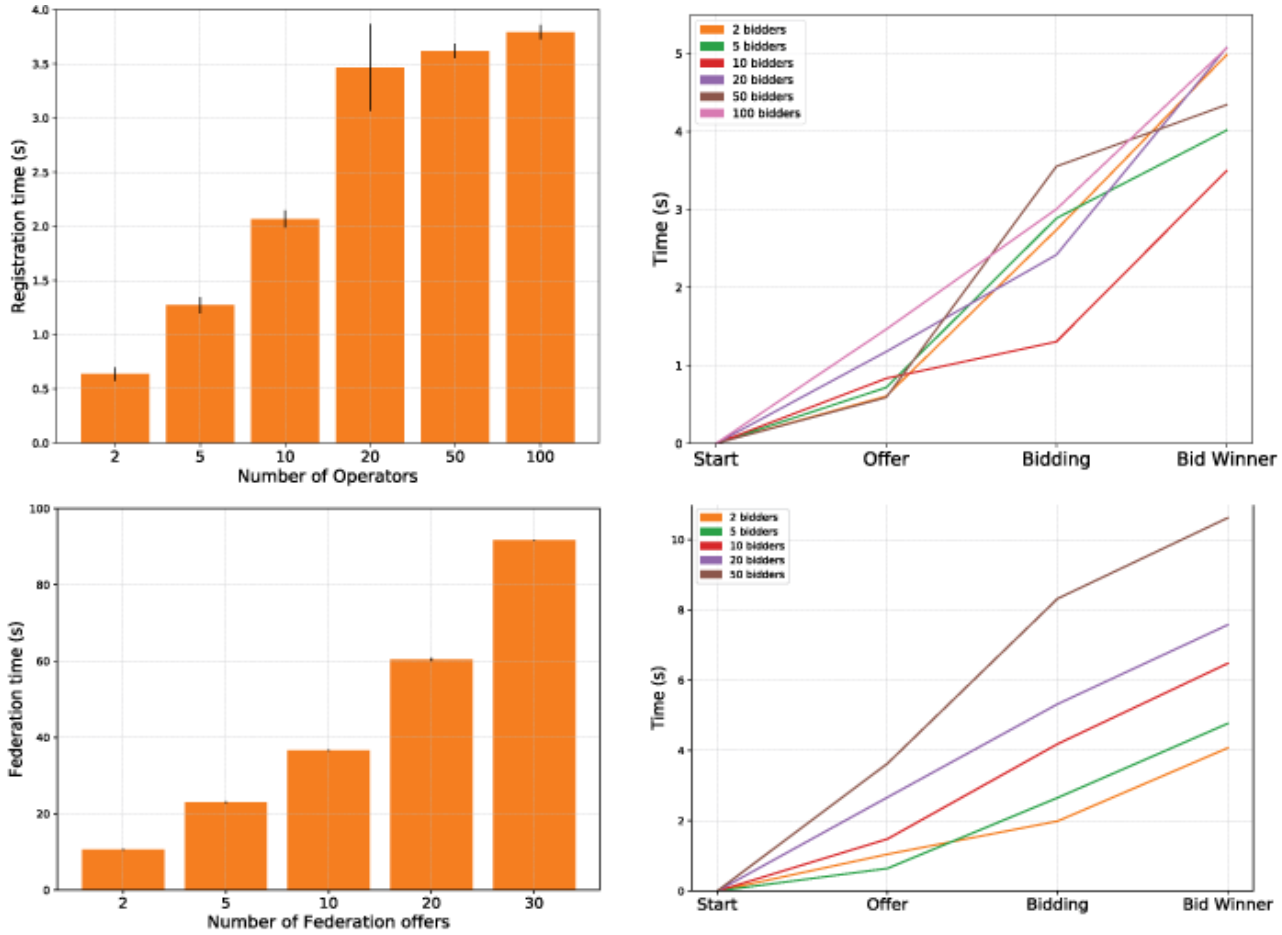


FIGURE 62: DLT FEDERATION RESULTS

4.7 SLA Management via Scaling Requests of Composite Network Services implying network service federation

This section presents the validation and assessment of the procedure along with the enhanced interface handling the auto-scaling of a nested NFV-NS deployed at the provider domain in an experimental multi-administrative domain infrastructure. This results in an interesting example of a scaling operation implying Network Service Federation (NSF) due to the required coordination and

interaction between the involved administrative domains after a "sudden change" in the structure of some parts of the composite NFV-NS as a reaction to a change in the service demands.

4.7.1 System architecture

Figure 63 presents the experimental setup under evaluation. This setup consists of two administrative domains (ADs), where each AD has an instance of the basic blocks of the 5Gr-architecture (i.e., 5Gr-VS, 5Gr-SO, 5Gr-RL and 5Gr-VoMs). The 5Gr-SO at the different AD platforms considers Open Source MANO (OSM) as Core MANO platform. AD1 uses OSM Release 6 and AD2 uses OSM Release 7. Different releases of OSM have been considered to validate the backward compatibility of the 5Gr-SO platform with external components while increasing the heterogeneity of the experimental setup. AD1 has three NFVI-PoPs, each one under the control of a dedicated Virtual Infrastructure Manager (VIM) based on Openstack Devstack Queens release. These NFVI-PoPs are interconnected by the depicted GNS3 [87] emulated transport network managed by an instance of the ONOS SDN controller acting as Wide Area Network Infrastructure Manager (WIM). AD2 has two NFVI-PoPs, also managed by its corresponding VIM (Openstack Devstack Queens release) instances. They are interconnected by another emulated GNS3 transport network, also managed by a dedicated instance of the ONOS SDN controller. At the data plane level, the ADs are linked through a VPN connection implementing the depicted Inter-AD Link. Another VPN connection is used at the control plane level to communicate the 5Gr-SOs supporting the SO-SO interface.

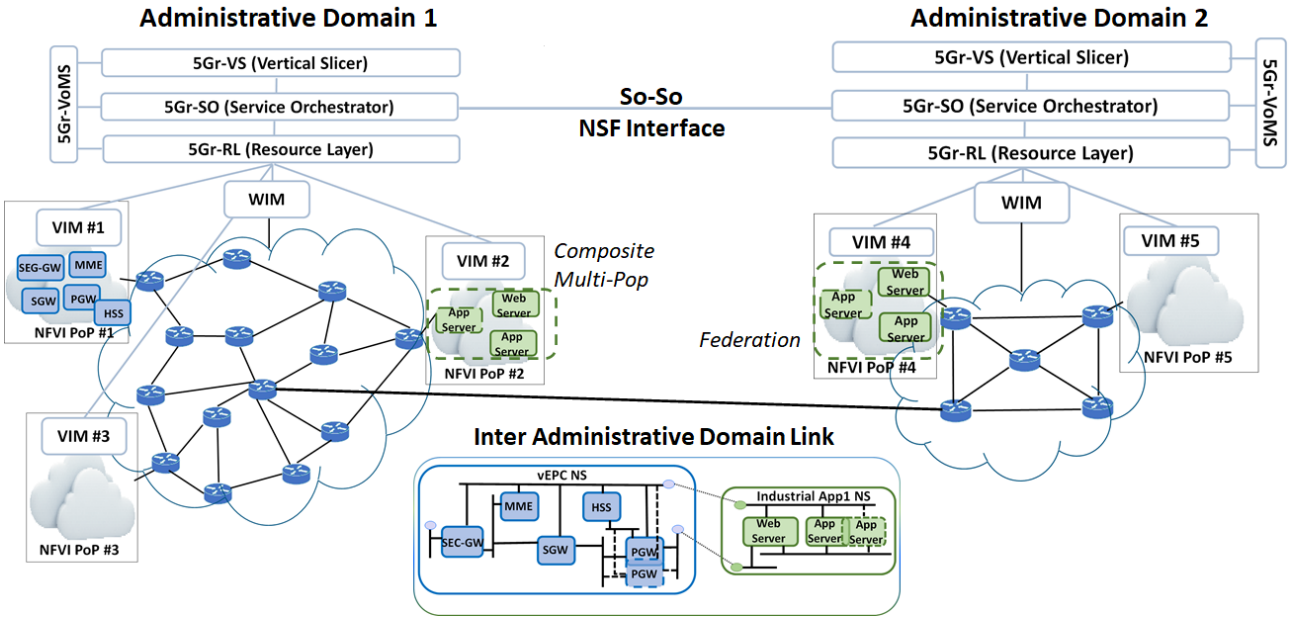


FIGURE 63: EXPERIMENTAL MULTI-ADMINISTRATIVE DOMAIN NETWORK SETUP

The considered composite NFV-NS is depicted in the bottom part of Figure 63. This composite NFV-NS presents two interconnected nested NSs, with multiple Instantiation Levels. These ILs allow the adaptation of the composite/nested NFV-NS structure fulfilling the NSD's SLAs and reacting upon dynamic network conditions and service demands. The first nested NFV-NS emulates a standalone

Non-Public Network (NPN) vEPC NS consisting of five VNFs. The second nested NFV-NS emulates an Industrial App NFV-NS, which consists of two VNFs.

4.7.2 Experimental results

The evaluation compares the average scaling time when an auto-scaling request is triggered by the nested industrial app NFV-NS to answer an SLA violation. On one hand, this SLA violation is bound to a CPU load increase in the App Server VNF caused by high amount of processing. This leads to trigger a scaling operation requesting an IL change to *scale-out* and thus adding a new instance of the App Server VNF. On the other hand, the SLA violation regards to the under-utilization of the CPU of the two available (after scale-out operation) App Server VNF instances due to reduced activity. This yields to a *scale-in* operation turning back to the initial IL with a single App Server VNF instance. The assessment of the scaling operation implying NSF is performed comparing the following deployments:

i) The composite NFV-NS has been instantiated between two different NFVI-PoPs of **a single AD** (labelled as *Composite Multi-Pop*), or
ii) between two different NFVI-PoPs managed by **different ADs** (labelled as *Federation*).

In both cases, the NPN vEPC nested NFV-NS is deployed in NFVI-PoP#1 of AD1, while the industrial app nested NFV-NS has been deployed in NFVI-PoP#2 or NFVI-PoP#4, as depicted in Figure 63. The experiments have been repeated ten times.

Table 19 shows the average scaling time (and standard deviation) required to *scale-out* the nested Industrial App NFV-NS. The total average time required to perform this operation in the *Federation* deployment is of 38.535±0.429 seconds.

TABLE 19: NESTED NFV-NS SCALE-OUT OPERATION PROFILING

5Gr-SO Entity	Federation (sec)	Composite Multi-Pop (sec)
5Gr-SOEp	6.409±0.174	0.035±0.001
5Gr-SOEc	23.736±0.174	40.331±1.64
5Gr-CROOE	8.390±0.429	5.747±0.414
Total	38.535±0.429	46.114±1.446

The different operations performed during this process have been grouped into three main groups considering the sub-modules in the 5Gr-SO responsible for them. For the federation case, this grouping also includes the elapsed time in the sub-modules of the different ADs.

The first group is the *5Gr-SOEp* time. This considers the time at the SOEp module/s to process the corresponding scaling requests, retrieve and update the information at the DBs, and the time needed by the consumer domain (AD1 of Figure 63) to verify that the nested NFV-NS at the provider domain (AD2 of Figure 63) has been successfully scaled. This is done through a periodical polling operation, and it is the main contributor to the 5Gr-SOEp time. The other mentioned components of the 5Gr-

SOEp are in the order of milliseconds. In this evaluation, the periodical polling was set to 10s to reduce the number of interactions between 5Gr-SOs since the time required to create/remove a VNF instance is in the order of tens of seconds.

The second group, the *5Gr-SOEc* time accounts for the time required by the SOEc sub-module to manage the actual scaling operation of the nested NFV-NS with the rest of modules of the 5Gr-SO. This is the most time-consuming operation in the whole scaling process and most of it is devoted to the interaction between the Core MANO platform (OSM software) and the VIM to create the virtual machine (VM) for the new instance of the server VNF. It is worth noting that the experienced value is bigger in the *Composite Multi-PoP* than in the *Federation* case, having a definitive impact in the total scaling time for the *Composite Multi-PoP* case that counteracts the effect of the polling operation mentioned previously. After a deeper analysis of experiment logs, we confirm this effect because of the different releases of the OSM MANO platform used in the different ADs. The newer release (OSM R7) used in AD2 presents a better procedure to create the VM when interacting with the VIM when compared to the older considered release (OSM R6).

Finally, the third group is the *5Gr-CROOE* value, representing the time required to determine the new inter-nested connections and its configuration at the different ADs by contacting with the corresponding RL modules. Out of the total value of the 5Gr-CROOE time of the *Federation* case (8.390±0.429s), around 72% corresponds to the time required to update the inter-nested connections at the AD1 transport network, which is more complex than the one of AD2. Approximately, this 72% of the *Federation* case is similar to the *Composite Multi-PoP* value (5.747±0.414s) because the amount of transport elements to configure from NFVI-PoP#1 and the Inter-AD link is similar to those between NFVI-PoP1 and NFVI-PoP2 depicted in Figure 63.

TABLE 20: NESTED NFV-NS SCALE-IN OPERATION PROFILING

5Gr-SO Entity	Federation (sec)	Composite Multi-Pop (sec)
5Gr-SOEp	5.362±2.386	0.035±0.001
5Gr-SOEc	21.784±2.455	23.129±2.594
5Gr-CROOE	3.218±1.364	2.327±0.900
Total	30.365±5.036	25.492±3.001

Table 20 shows the results when considering the *scale-in* operation, when the nested Industrial App NFV-NS turns to its initial structure with just a single instance of the App server VNF. The average value of this operation is of 30.365±5.036s in the *Federation* case and of 25.492±3.001 in the *Composite Multi-Pop* case. As observed in [88], the *scale in* operation (i.e., resource de-allocation) is performed faster than the *scale out* operation (i.e., resource allocation). In the *scale in* operation, as expected, the *Federation* case attains a higher value than the *Composite Multi-Pop* case. The time difference mainly comes from the mentioned polling operation time done in the consumer domain, contributing also to more variability in the measurements. In both cases, the Core MANO platform obtains a similar performance when removing the previously added App server VNF virtual machine. Regarding the update (deletion) of inter-nested connections considered in the 5Gr-SOEc component,

it is faster than the creation of new ones, but it presents a similar trend when compared with the *scale out* case and the different considered cases.

In summary, this experimental study links Innovation 6 and Innovation 4 by providing SLA management in the form of scaling actions during run-time to network slices/NFV-NSs deployed in multiple administrative domains using network service federation procedures. This increases the 5Gr-SO automatic network service management capabilities and provides compatibility with self-adaptation mechanisms. The proposed scaling process implies coordination between ADs, having an impact in the overall procedure, but it is still in the order of seconds, in-line with the expected KPIs. However, the most time-consuming operations are those related to the actual allocation/release of underlying resources (VMs and network connections). Multiple factors impact in the measurements such as the used hardware, the adopted virtualization technique of computing resources (VMs), and the amount of transport forwarding elements to be configured to deploy the interconnection between VNFs located at remote NFVI-PoPs. Additionally, this experimentation also confirms the impact of the different releases of OSM in the measured time. All these aspects need to be considered in the SLA management of such generic E2E slice deployments.

4.8 Next-generation Radio Access Networks

In this section, we evaluate vrAIn, our O-RAN use case introduced in Section 3.1.7.2, through two methodologies: (i) doing experiments in a small proof-of-concept prototype with up to two vRAPs, and (ii) simulating a real-life large-scale network in Romania. The details of our experimental evaluation can be found in [91].

4.8.1 Experimental Proof-of-Concept

A video demonstration of our experimental proof-of-concept can be found in [39].

We set up two vRAPs in a i7-5600U compute node limiting its capacity to a single CPU core, i.e., both vRAPs (“vRAP1” and “vRAP2”) have to compete for these resources during peak periods, and use the dynamic load and SNR patterns shown in the first two top subplots of Figure 64.

The remaining two bottom subplots depict the temporal evolution of vrAIn’s CPU policy and radio policy. First, we can observe that vrAIn distributes the available CPU resources across both vRAPs following their context dynamics - equally between them when the contexts are similar. More importantly, we note that vrAIn reduces radio policies when the aggregate demand is particularly high to minimize decoding errors due to CPU deficit.

A larger set of tests and benchmarks of our experimental prototype can be found in [38].

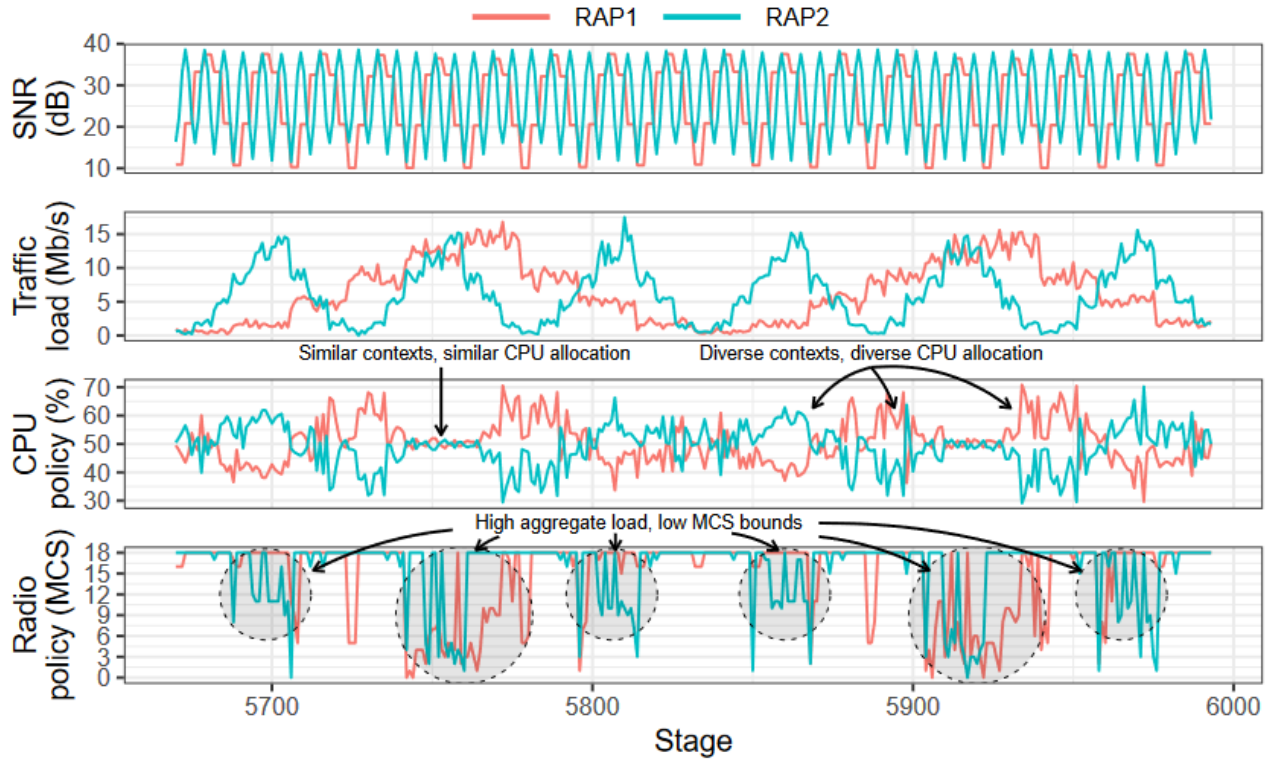


FIGURE 64: VRAN WITH 2 VRAPS. VRAIN ADAPTS THE RADIO SCHEDULING POLICIY TO MINIMIZE DECODING ERRORS (WHICH ARE NEGLIGIBLE AND HENCE NOT SHOWN)

4.8.2 Simulations

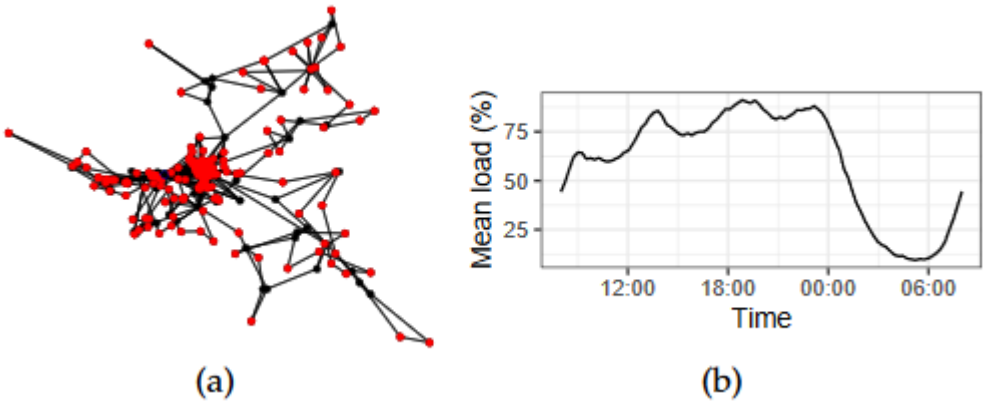


FIGURE 65: (A) DEPLOYMENT OF AN OPERATIONAL RAN IN ROMANIA. RED AND BLACK DOTS REPRESENT, RESPECTIVELY, RADIO SITES AND BACKHAUL AGGREGATION NODES. (B) TRAFFIC LOAD PATTERN OVER A PERIOD OF 24H OF A REGULAR WEEKDAY.

To assess the performance attained by in a real-world scenario, we simulate over a production RAN deployment in Romania, with 197 access points distributed as shown in Figure 65-a). As we can observe from the figure, there is a higher density of RAPs in the datacentre (a big city) and the RAN is sparser by the outskirts (covering mostly highways and small commuter suburbs). Our simulator follows the 3GPP guidelines for LTE performance evaluation, and its parameters are detailed in [38].

We simulate one regular week using synthetic traffic patterns, which emulate the behaviour of real RANs at scale (see [38] for details). To simplify the analysis, in the following we focus on a 24-hour period during weekdays—with the traffic profile (relative to the capacity of the system) shown in Figure 65-b); but similar conclusions can be obtained from any other day. We further assume that the aggregate computing capacity of the whole vRAN is dimensioned to the minimum amount of CPU resources required such that no due to CPU deficit occur during the load peak of the day when not using vrAIn, which we refer to as "100% provisioning". We also study the behaviour of when the system is under-provisioned to 70% and 85% of that computing capacity, which enables capital cost savings.

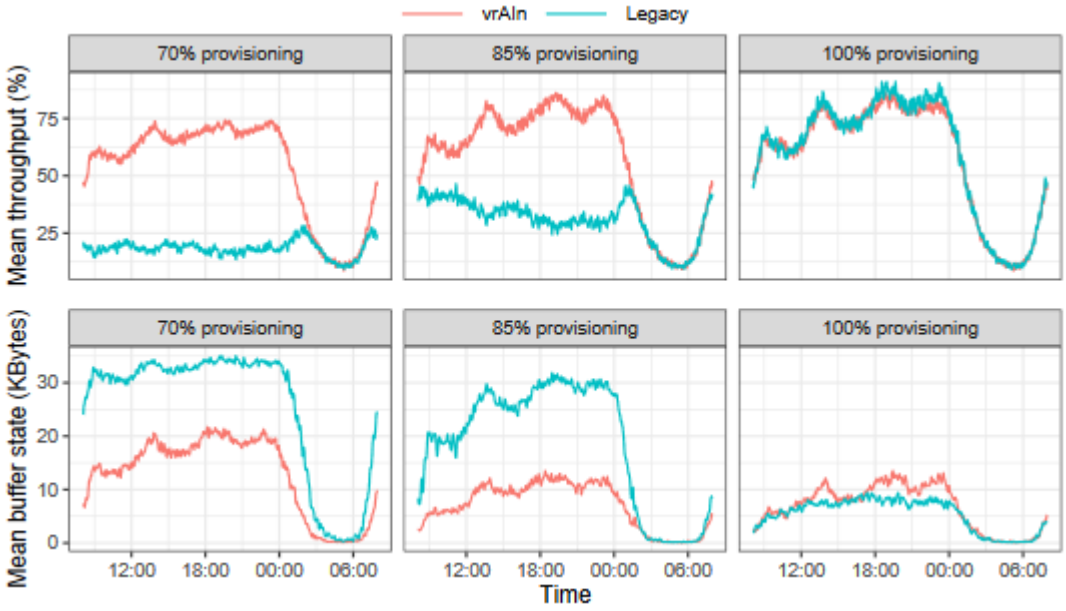


FIGURE 66: PERFORMANCE EVALUATION OVER TIME FOR A COMPUTING CAPACITY DIMENSIONED FOR THE PEAK LOAD ("100% PROVISIONING") AND FOR 70% AND 85% OF THAT COMPUTING CAPACITY. MEAN THROUGHPUT (TOP). MEAN BUFFER OCCUPANCY (BOTTOM).

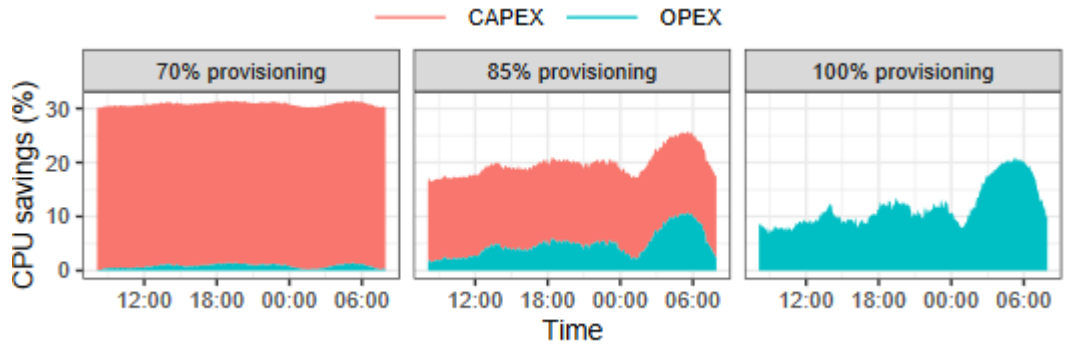


FIGURE 67: VRAIN'S SAVINGS RELATIVE TO A STATIC COMPUTING-AGNOSTIC APPROACH PROVISIONED WITH SUFFICIENT CPU RESOURCES FOR THE PEAK DEMAND.

Let us first focus on the evolution of throughput performance, buffer states and cost savings when the computing capacity is provisioned to accommodate the peak load (right-most plots in Figure 66 and Figure 67, labelled as "100% provisioning"). From Figure 66 we observe that achieves roughly

the same throughput performance and slightly higher buffer sizes (up to 5%) than "Legacy". This is explained because trades off this slightly higher delay for substantial OPEX savings. This is shown in Figure 67 (right-most plot, labelled as "100% provisioning"), where achieves between 10% and 20% of OPEX savings. Note however that this difference in delay vanishes when is configured to favour performance over OPEX savings (results omitted here for the sake of space).

We now analyse the case where we impose under-provisioning to obtain aggressive CAPEX gains by reducing the availability of computing capacity to 85% and 70% relative to the dimensioning strategy discussed before, and labelled as "85% provisioning" and "70% provisioning", respectively, in Figure 66 and Figure 67. The evolution of throughput and buffer states in Figure 66 for these two scenarios (left-most and middle plot, respectively) make it evident how enables aggressive CAPEX savings while retaining high performance gains when compared to legacy computing-agnostic approaches. Specifically, provides up to 50% and 55% throughput gains over our legacy scheme when, respectively, the computing capacity is under-provisioned to 85% and 70% of the peak load, in average, and during the time of peak demand (between 18:00 and 00:00). The reason lies on the fact that vrAIn's ability to optimize jointly radio and computing policies allows to better accommodate the demand along the time domain, trading off delay for near-zero due to a deficit of computing capacity. In contrast, legacy's agnostic behaviour with respect to the availability of computing capacity yields substantial throughput loss due to a high rate of during instantaneous peak demands. This produces a cascade effect causing large amounts of time wasted in re-transmissions and rendering even larger perceived latency: up to 160% and 73% higher buffer occupancy over for 85% and 70% of computing provisioning, respectively, in average, and in the same period of peak demand.

A final remark is that adapts, i.e., maximizes performance, to constrained computing environments. This can be observed by comparing the performance indicators of both "70% provisioning" and "85% provisioning" to those shown for "100% provisioning". In particular, has no loss in throughput performance for "85% provisioning" and only up to 10% throughput loss for "70% provisioning", in average, and during the same period of peak demand discussed earlier. This contrasts with the substantial throughput loss attained by our legacy approach: up to 65% and 80% of throughput loss in those same conditions. In terms of latency, only suffers from 1% increase in mean buffer occupancy in the case of "85% provisioning" relative to the buffer occupancy for "100% provisioning" (in contrast to the 282% increase of the legacy approach), and 73% increase in mean buffer occupancy with 75% under-provisioning (in contrast to the 337% increase of the legacy approach).

4.9 Smart Orchestration and Resource Control

Within the following three subsections, the performance of each individual algorithm among the three topics mentioned in Section 3.2.1 is evaluated: (1) Smart orchestration, (2) Resource abstraction and allocation, and (3) Dynamic profiling mechanism. Note that all algorithms are evaluated in a self-contained manner, and thus their respective environment configuration and simulation/emulation parameters can be found in the corresponding subsection.

4.9.1 Smart Orchestration

4.9.1.1 GA-based SFC placement

An extended set of diverse experiments have been conducted, in order to examine the quality of solutions produced by the proposed algorithm, under different network set ups and algorithm configurations. For our simulation scenarios a partial mesh network topology is considered. The network topology comprised 20 physical hosts, each one equipped with 2 VMs, 35 physical links and 8 network L2/3 switches. The experiments were conducted on an Intel(R) Core(TM) i7-6800K CPU of 3.4 GHz speed with 8 GB of RAM, running Ubuntu 16.04. Table 21 includes all GA's configuration parameters.

TABLE 21: GA CONFIGURATION

Parameter	Configuration
Population size	50
Generations	200
Initial threshold	5
Elites	40
Crossover rate	0.5
Mutation rate	0.2
Mutagenesis rate	0.2

For each round of results, the Mean Fitness Value (MFV) and Mean Execution Time (MET) metrics are derived, over 15 different experiment runs per experiment section.

The quality of the solution produced by the GA framework is compared against the optimal one, as provided by the MILP model under a brute force search approach. This comparison considers an increasing number of required SFCs and VNFs per SFC in the left part of Figure 68-a) and Figure 68-b). For almost all scales considered, the solution of the proposed algorithm coincides with the optimal one (MILP) (left part of Figure 68-a)); Even in the last case (highest SFC and VNF scale equal to [10, 7]), GA's solution is almost identical to the optimal. As shown in the left part of Figure 68-a), the proposed GA model presents low time complexity, for all scales considered.

As a next step, the proposed scheme is juxtaposed with the MILP model under increasing scale of the topology, in terms of number of physical hosts (in the eight part of Figure 68-a) and Figure 68-b)). Both present an identical performance in almost all topology scales selected, in terms of MFV. The proposed algorithm produces a solution almost equal to the optimal solution (produced under the MILP model) even when the search space increases (higher topology scales). Under the proposed algorithm, the observed MET is comparable to that under the smaller search space (lower topology scales), which is clearly not the case under the MILP approach where the observed MET seems to increase exponentially with the scale.

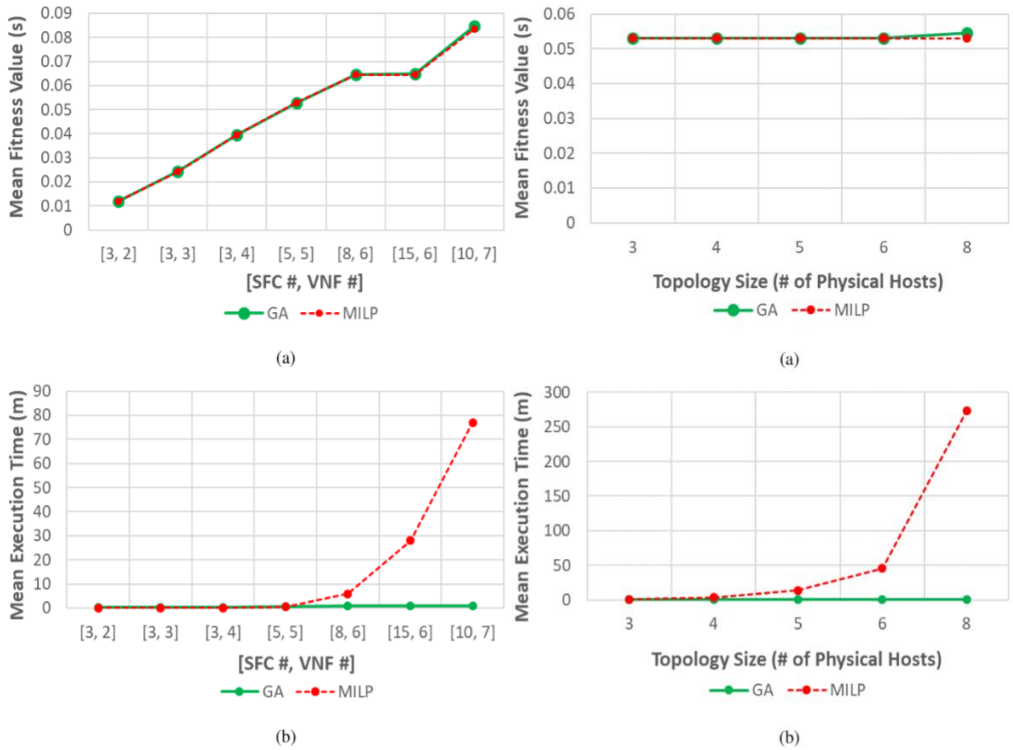


FIGURE 68: GA AND MILP COMPARSSION IN TERMS OF (A) MEAN FITNESS VALUE AND (B) MEAN EXECUTION TIME, FOR DIFFERENT TOPOLOGY SIZES IN TERMS OF SDC/VNF NUMBER (LEFT) AND THE NUMBER OF AVAILABLE PHYSICAL HOSTS (RIGHT).

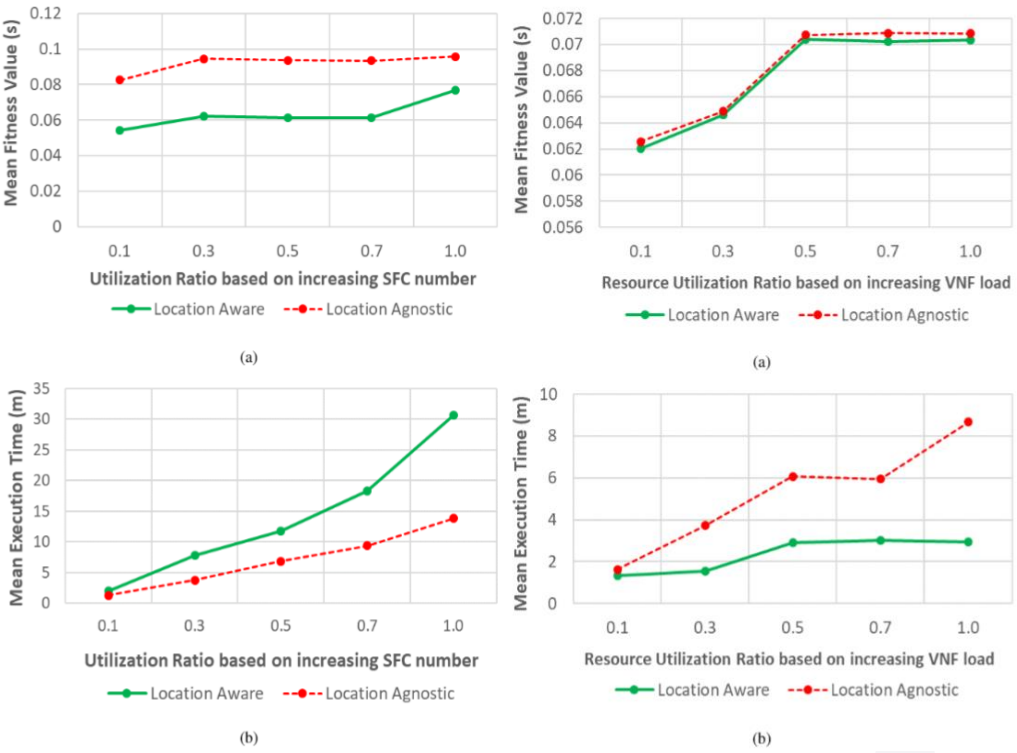


FIGURE 69: LOCATION-AWARE VERSUE LOCATION-AGNOSTIC COMPARISON FOR INCREASING UTILIZATION RATIO, IN TERMS OF THE NUMBER OF SFCS TO BE DEPLOYED (LEFT), AND VNF REQUESTED LOAD (RIGHT)

In the final part of our evaluation, the proposed scheme is evaluated when the introduced location-aware optimization method is applied. All 4 subfigures in Figure 69 illustrate the effectiveness of the location-aware method of the proposed scheme. The solutions developed with and without (location-agnostic) the application of the location-aware method are compared for different Utilization Ratios, in terms of increasing SFC number and VNF requested load. In both scenarios, location-aware optimization method results in better quality solutions in terms of MFV.

This shows the feasilbility of the GA-based SFC placement scheme for the 5Growth End-to-End Service Platform towards smart resource orchestration, aiming at minimizing the end-to-end delay of ultra-low delay industrial network operations, via also exploiting location aware information. A comprehensive problem formulation as a MILP optimization problem is presented and a low-complexity GA-based solution to the SFC placement problem is derived, incorporating an early stopping criterion, as well as the proposed location-awareness and solution filtering optimizations. The proposed GA-based approach is shown to present a practically useful trade-off between the closeness of the derived solution to the optimal and the time needed to obtain this solution.

4.9.1.2 VNF Autoscaling

In order to validate the performance of the proposed MLP model, the ns-3 discrete-event network simulator [92] was used. The deployed scenario, with total duration equal to 12000 sec, assumes 12 UEs connected to a specific gNB, running two simulated eMBB services (Fast Browsing (FB), UHD Video Streaming (UHDVS)), with different data rates and traffic volumes. The base station is interconnected with a MEC server, hosting one Virtual Machine (VM). This study applies the proposed scheme to a part of an eMBB service, represented by a pre-configured VNF type (see Table 22). Service-related measurement traces are logged in the form of time-series information, while the logging frequency is set to 1 second. The time period is set to 5 seconds.

TABLE 22: SERVICE TEMPLATE

Parameter	FB	UHDVS
VNF type	vnf_fb	vnf_uhdvs
VNF throughput (packets/sec)	250	150
Packet Interval (ms)	20	10
Packet Size (bytes)	300	1000

Since, ns-3 currently does not support the modelling of VNF instances, the correlation between the simulated data and ILs is modelled under a threshold-based logic. More specifically, it is assumed that the traffic load (in packets) is equally distributed among the VNF instances running on a VM host. Each VNF instance can handle specific throughput (packets/sec). Every 1 second, given the traffic load in packets (produced by ns-3), the CPU load of each VNF instance is defined by the division of the traffic load by the VNF throughput. In addition, every second, the described algorithm verifies whether the average CPU load of all running VNF instances is greater than a predefined upscale threshold thr_{up} (set to 0.8) or less than/equal to a predefined downscale threshold thr_{dw} (set to0.5) of the VNFs' throughput in order to upscale or downscale to the appropriate IL (by dividing

the CPU load of each VNF instance by $thr_{up/dw}$). The resulted ILs comprise the set Y_s that is provided as a new extracted label set to the respective MLP model. To increase the performance of the MLP models, the Stochastic Gradient Descent (SGD) with momentum was selected, with learning rate equal to $\eta = 0.001$. All models used L2 (weight decay) regularizers with regularization factor of 10^{-4}. The split ratio for the dataset is set to 0.8. Figure 70 illustrates the two timelines of the predicted and the real (ground truth) labels of the test set for both VNF types. As it can be inferred, the predicted and actual timelines present high similarity for both VNF types. The latter is validated by Table 23, which depicts the performance of the two MLP models using accuracy and f1-score as metrics. More precisely, the accuracy is equal to 88.98% and 90.38% respectively, while the f1-score amounts to 88.89% and 90.14% for vnf_fb and vnf_uhdvs respectively.

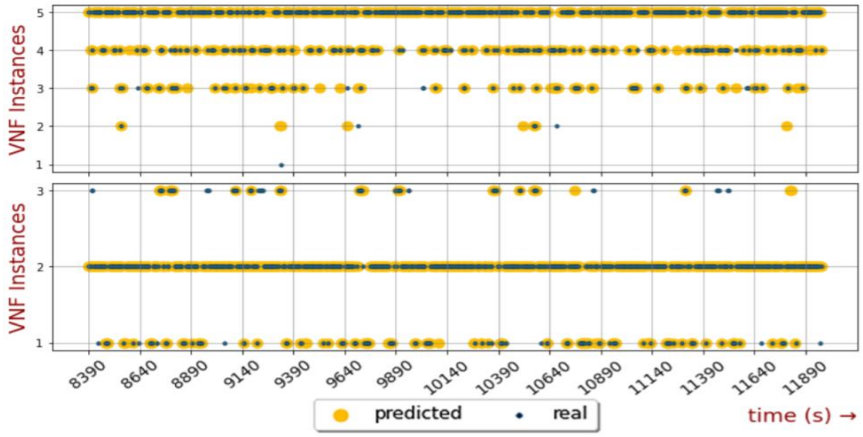


FIGURE 70: PREDICTION – GROND TRUTH TIMELINE FOR VNF TYPES [VNF_FB (UP), VNF_UHDVS (DOWN)]

TABLE 23: MODEL PERFORMANCE

VNF Type	Accuracy	F1-Score
vnf_fb	88.98 %	88.89%
vnf_uhdvs	90.38 %	90.14%

The proof-of-concept experiment proposed a Deep Neural Network (DNN)-based scheme using a MLP model, trained in the 5Gr-AIMLP, for proactively service scaling through the interaction between the 5Gr-AIMLP and the 5Gr-SO and according to the respective network requirements and real-time traffic load. A simulation scenario using ns3 discrete event simulator was deployed, modeling two different eMBB services, in order to validate the performance of our model. Simulation results prove the validity of our solution providing high accuracy and f1-score values for both service types.

4.9.1.3 AI-Driven scaling on of C-V2N Services

In this section, we show the performance of the AI-Driven scaling solutions for C-V2N services proposed in Section 3.2.1.1.3. Below Figure 71 and Figure 72 refer performance of several evaluated approach: (*i*) CNST as the constant number of CPUs benchmarking solution; (*ii*) PI as the Proportional

Integer (PI) controller; (*iii*) RL as the Q-learning agent tackling the scaling as a reinforcement learning problem; and (*iv*) LSTM as the long-short term memory based neural network, that forecasts the future load to accordingly scale up/down the number of CPUs.

Specifically, Figure 71 and Figure 72 illustrate the evaluation of the number of CPUs over time. The x-axis has a frequency of 5 minutes, and both figures are related, as higher number of CPUs result on lower maximum loads in the CPUs (see both the RL and CNST approaches). However, it is desired that resources are not underused, that is, it would be desirable that the CPUs are always at their maximum load. On the other side, we do not want that the required workload $W(t)$ cannot be processed by the current number of CPUs $N(t)$, hence, the reward function that measures the goodness of each control agent (*i*) wants CPUs being as much loaded as possible; and (*ii*) penalizes not being capable of processing the existing workload.

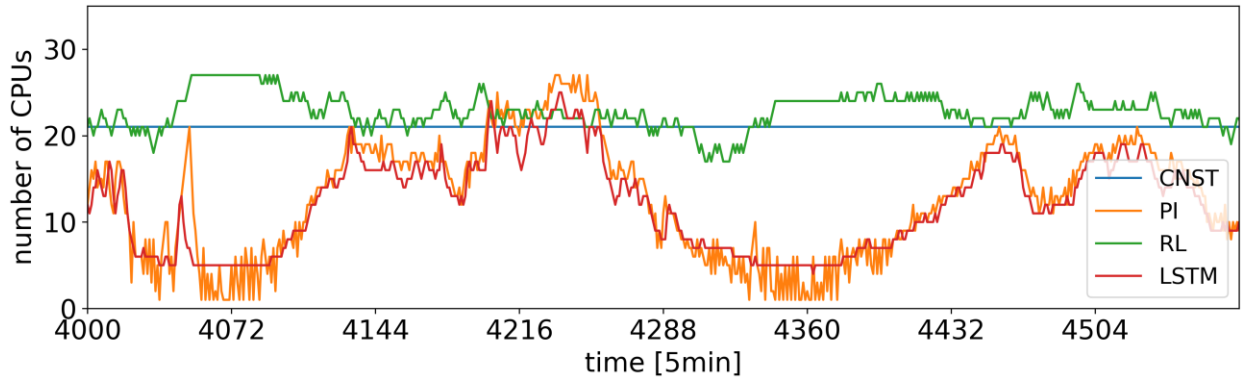


FIGURE 71: C-V2N SCALING - EVOLUTION OF NUMBER OF CPUS

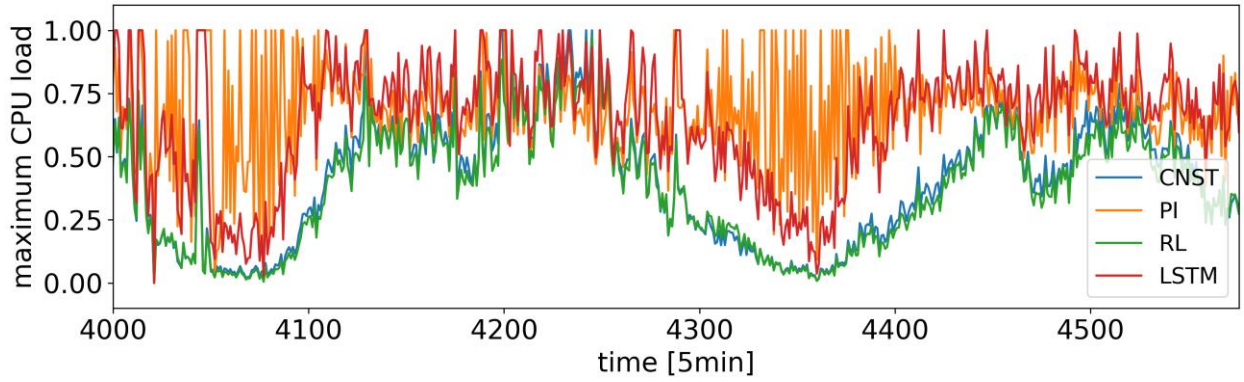


FIGURE 72: C-V2N SCALING - EVOLUTION OF CPU LOAD

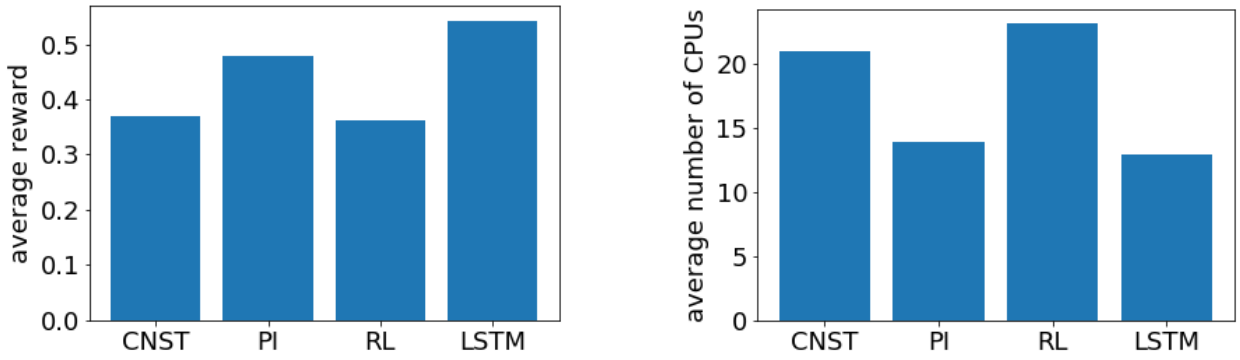


FIGURE 73: AVERAGE REWARD METRIC (LEFT) AND AVERAGE NUMBER OF CPUS METRIC (RIGHT)

Then, the left-hand part of Figure 73 shows the average reward experienced by the analysed scaling agents over the time interval shown on both Figure 71 and Figure 72. As the reader will notice, the LSTM outperforms all the analysed solutions and achieves the highest reward, this is because it has the lowest average number of CPUs (see the right part of Figure 73). Nevertheless, the low number seems to be enough to meet the workload without pit falling in under-dimensioning the number of CPUs for the incoming workload.

4.9.1.4 Slice Sharing at 5Gr-VS Arbitrator

Here we present a sample of the preliminary result we have obtained through the implementation of the slice sharing algorithm at the VS Arbitrator (see Section 3.2.1.1.4), with the latency classification provided by an ML model trained through the 5Gr-AIMLP. We mainly focused on the optimization of vCPU consumption and consider three different generic services composed of the following VNFs: $s_1 = (v_a, v_b, v_c)$; $s_2 = (v_c, v_d, v_e)$; $s_3 = (v_d, v_f)$. The services have the following VNFs in common: v_c for s_1, s_2 and v_d for s_2, s_3. All the services require to be deployed at the edge of the infrastructure.

During the experiments, the CPU consumption was measured for 1200 second under two different workloads, namely high and low, with average number of instances per service equal to 9 and 1, respectively. Figure 74 presents the consumption of virtual cores when the slice sharing algorithm is in place and when no slice sharing is enabled. Despite the small-scale scenario we considered, the results already show an improvement in the resource usage when slice sharing is allowed.

It is worth emphasizing that a key parameter of the algorithm is the latency class configuration (i.e., the number and width of each latency class) provided by the trained ML model. This parameter mainly depends on the number of active services and their target latency, and as they vary, the latency class configuration enables the algorithm to make better decisions.

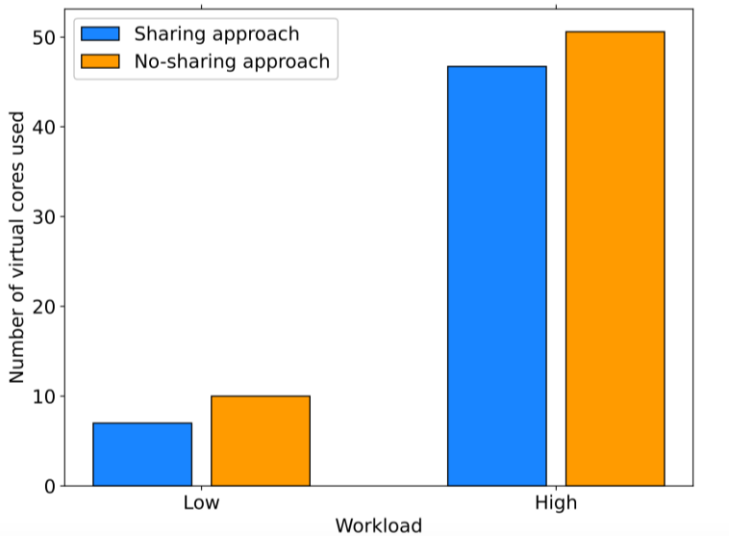


FIGURE 74: AVERAGE NUMBER OF VIRTUAL CORES USED: SLICE SHARING VS. NO-SHARING

4.9.2 Resource Abstraction and Allocation

4.9.2.1 5Gr-SO-5Gr-RL: Resource Abstraction and Allocation Algorithms

This section experimentally tackles the scalability evaluation of both CSA and InA modes described in Section 3.2.1.2.1 under two NFVI scenarios considering diverse the WAN size in terms of number of nodes and links.

4.9.2.1.1 Considered WAN scenarios and main assumptions

Figure 75 depicts the two considered NFVIs. Each NFVI is formed by a single domain formed by three NfviPops. The WAN infrastructure interconnecting the NfviPops is either: a) a Spain core WAN formed by 14 packet-switch nodes and 22 bidirectional (see Figure 75(a)); or b) a small metro network with 5 packet switch nodes and 8 bidirectional links (depicted in Figure 75(b)). For both WAN topologies, every link has a transport capacity of 1Gb/s where the associated delay (in ms) is labelled on the link. The connectivity between the NfviPop termination point (i.e., gateway) and attached packet switch (e.g., NfviPop1 and S1 in Figure 75(b) provides a net data rate of 1Gb/s with negligible link delay. In both NFVIs scenarios, a centralized WIM (i.e., SDN controller) takes over of the WAN programmability with emulated flow configuration. Received NSReqs are handled by the 5Gr-SO (via a MANO OSM). Three different VIMs (OpenStack) manage the respective NfviPops. The VIMs and WIM instances are coordinated by the 5Gr-RL to de-/allocate the NFVI resources. The 5Growth stack components (i.e., 5Gr-SO, 5Gr-RL, RL RA Server, VIMs and WIM) are installed over different Linux-based servers at the CTTC 5G end-to-end platform.

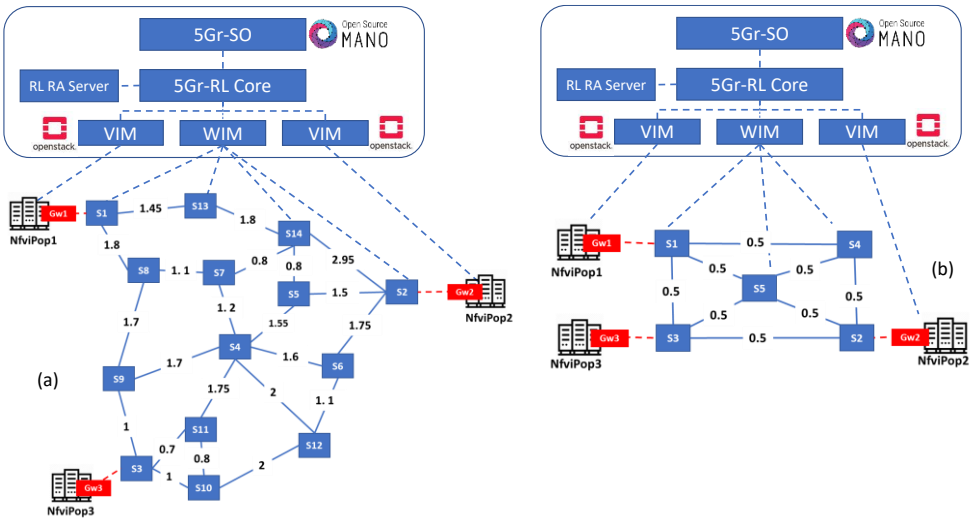


FIGURE 75: (A) SPAIN CORE WAN; (B) SMALL METRO WAN

To conduct the validation and scalability evaluation, we rely on an NFV-NSReq (Figure 76) based on the eHealth use case investigated in the 5G-TRANSFORMER project [15]. Such NSReq requires a NS descriptor formed by six VNFs: 5 VNFs (i.e., MME, PGW, HSS, SGW and SEC-GW) which do implement the entities of a virtualized evolved packet core (vEPC); the remaining VNF is a server VNF working as a back-end entity for a specific eHealth application. Moreover, the targeted NFV-NSReq encompasses six VLs supporting the different interfaces present in the EPC system, i.e., SGi, S5, S8, S6b, S1 and a VL shared between all the VNFs for the management traffic. The VLs are shown in Figure 76 with solid lines, whilst the numbers represent the NFV-NS service access points. In this section, we are not interested in the internals of the application, but only on its complex NFV-NS topology. It is assumed that the 5Gr-SO placement function distributes the set of VNFs among the three available NfviPops. Specifically, the SEC-GW and SGW VNFs are deployed at NfviPop1, the rest of vEPC entities (i.e., MME, PGW and HSS) are allocated at NfviPop2, and the server VNF is placed at NfviPop3. As a result of this VNF distribution, the 5Gr-SO requires to the 5Gr-RL 15 inter-NfviPop connections (i.e., LLs).

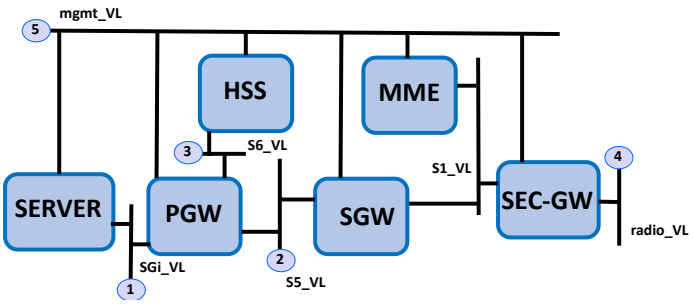


FIGURE 76: STRUCTURE OF THE NFV-NS UNDER EVALUATION

Exclusively for the CSA operational mode and without loss of generality, Table 24 details the offered LLs for both Spain and Small WANs considering three defined Gold/Silver/Bronze connectivity service types/classes. Each service type defines a pair of guaranteed parameters: maximum bandwidth and delay. Observe that depending on the WAN size, high performance connectivity service types (e.g., Gold) could not be offered because the maximum permitted latency/delay between a pair of

NfviPops is exceeded. In the considered WAN scenarios, for the Small one, all three defined connectivity service types are allowed for any NfviPop pair. Nevertheless, in the Spain WAN, this does not occur.

TABLE 24: SUPPORTED CONNECTIVITY SERVICE TYPES AND LLS FOR THE CONSIDERED WAN SCENARIOS IN THE RA CSA APPROACH

WAN Scenario	NfviPop connectivity	Supported Connectivity Service Types	Connectivity Service Type Features [Max. Bw, Max Latency]
Spain WAN	NfviPop1 <-> NfviPop2	Bronze	**Gold:** [100 Mb/s, 3ms] **Silver:** [50 Mb/s, 5ms] **Bronze:** [25 Mb/s, 10 ms]
	NfviPop1 <-> NfviPop3	Silver & Bronze	
	NfviPop2 <-> NfviPop3	Bronze	
Small WAN	NfviPop1 <-> NfviPop2	Gold, Silver & Bronze	
	NfviPop1 <-> NfviPop3	Gold, Silver & Bronze	
	NfviPop2 <-> NfviPop3	Gold, Silver & Bronze	

4.9.2.1.2 Results and discussion

Table 25 shows the numerical results obtained when adopting either the InA or CSA operational modes for both the Spain and Small WAN topologies when deploying the described NFV-NSReq. These results are given in terms of:

- Number of RL RA algorithm executions (either for abstraction or expansions functions).
- The elapsed RA InA and CSA algorithm computation time.
- The number of abstracted LLs exposed by each operational mode to the 5Gr-SO.
- The abstraction information time.
- The time required by the 5Gr-RL to successfully deploying a received LL over the WAN path.

The goal is to evaluate and compare the scalability (using the two considered WAN size infrastructures) attained by InA and CSA operational modes. According to the assumed VNF distribution through the three available NfviPops (detailed above) 15 inter-NfviPops connections (LLs at the 5Gr-SO context) are required to be deployed. In the 5Gr-RL, these inter-NfviPop connections are then rolled out over WAN paths. It is worth highlighting that a selected LL by the 5Gr-SO may accommodate more than one inter-NfviPop connection. In the InA operational mode, this means that such inter-NfviPop connections are set up over the same WAN path. Nevertheless, this may not happen in the CSA mode. That is, since the CSA mode triggers independent expansion WAN path computations for each inter-NfviPop connection, it is likely that even if two (or more) inter-NfviPop connections are bound to the same LL (e.g., Gold LL between a given NfviPop pair), the resulting WAN paths are different.

In the InA operational mode, the whole abstraction computation algorithm is executed once. In other words, the output of the InA algorithm is formed by all the LLs (for all the possible NfviPop pairs) that eventually are exposed to the 5Gr-SO. As expected, the considered WAN size notably impacts on the RA InA abstraction computational time: 290 ms and 77 ms for the Spain and Small WAN topologies, respectively. Indeed, the larger the WAN topology is, the more connectivity options

between NfviPop pairs can be explored. Thus, this leads to increase the abstraction computation time. On the other hand, the RA algorithm for the CSA operational mode is exclusively triggered for the expansion WAN path computation. For the targeted NFV-NSReq requiring 15 inter-NfviPop connections, the RA CSA expansion algorithm is thus triggered 15 times. For each of them, it is needed to: i) retrieve the WAN information, and ii) execute the RA algorithm for the selected LL supporting an inter-NfviPop connection. Therefore, the RA CSA algorithm time also depends on the WAN size. For the Spain WAN scenario, this computation takes around 22 ms, whilst for the Small WAN it requires 14 ms.

TABLE 25: COMPARISON OF INA AND CSA OPERATIONAL MODES

WAN Scenario	Number of RL RA Executions	RL RA time (ms)	Number of Abstracted LLs	Abstracted Info time (ms)	LL deployment time (ms)
Spain InA	1	290	18	2683	2595
Spain CSA	15	22	8	1859	4708
Small InA	1	77	18	2200	1498
Small CSA	15	14	18	1683	4486

The number of LLs advertised to the 5Gr-SO varies according to the adopted 5Gr-RL operational mode. In the InA approach, for both Spain and Small WAN topologies, the number of abstracted LLs are 18. This is obtained considering K = 3 WAN paths resulting on inferring 3 (unidirectional) LLs for every NfviPop pair. However, for the RA CSA approach, the number of abstracted LLs depends on the feasibility to support the defined connectivity service types/classes for each NfviPop pair. As shown in Table 24, the CSA approach for the Spain WAN scenario limits the number of exposed LLs to 8. As mentioned above, in the Spain WAN topology and for the defined connectivity service types, there are some NfviPop pairs where Gold and/or Silver service types cannot be offered since no WAN path fulfils the maximum end-to-end delay for those service types. Conversely, for the Small WAN topology, all the connectivity service types are offered for all the NfviPop pairs, resulting on 18 LLs exposed to the 5Gr-SO.

The abstracted information time is also a relevant metric to be discussed. For the sake of completeness, this time is defined as the total amount of time required by the InA algorithm to derive all the LLs that eventually constitute the abstraction information exposed to the 5Gr-SO. In general, regardless of the WAN scenario, the CSA approach attains lower time than the one obtained in the InA mode. This is because in the CSA mode no explicit abstraction path computation to derive the LLs is triggered. Focusing on the InA operational mode, again larger WAN infrastructures does increase the abstraction RA computation time, which in turn enlarges the total abstraction information time. Finally, the attained LL deployment time, i.e., deploying an inter-NfviPop connection over a feasible WAN path, results more time consuming in the CSA mechanism when compared to the InA approach. This behaviour is observed in both considered WAN topologies (e.g., in the Spain WAN, the LL deployment achieved by the CSA mode takes 4708 ms, whilst in the InA this time is 2595 ms). Indeed, as discussed above, in the CSA approach the deployment of every

inter-NfviPop connection over a selected LL entails: i) retrieving an updated view of the WAN infrastructure; ii) triggering the expansion WAN path computation for the inter-NfviPop connection; iii) conducting the packet flow configuration over the computed WAN path. However, in the InA approach, each inter-NfviPop connection to be rolled out over a selected LL has already determined the WAN path. Thus, there is no need to neither retrieving the WAN status nor executing a WAN path computation. This explains the reason why the CSA approach requires larger time when deploying an inter-NfviPop connection over a selected LL than in the InA approach.

Considering the above results, both InA and CSA operational modes have their own pros and cons. The InA approach requires a more complex and thus time-consuming abstraction mechanism for deriving the LLs than the CSA approach. On the other hand, the deployment of the selected LLs in the CSA approach needs to trigger (several) expansion WAN path computations which is not necessary in the InA operational mode. Moreover, we observe that the underlying WAN characteristics (i.e., number of nodes and links) impact significantly in the associated path computation functions (i.e., abstraction / expansion) realized by each operational mode. In general, the higher is the WAN size, the more time-consuming processes result both the abstraction computation (done by the InA approach) and the expansion path computation (made by the CSA approach). From a more qualitatively perspective, one can state that the InA approach allows exposing a more accurate abstraction view of the LLs to the 5Gr-SO. This allows the service provider accomplishing a more efficient selection of the virtualized resources when deploying a NFV-NSReq. On the other hand, the capability of the CSA approach to trigger dedicated expansion path computations for each selected LL may lead to achieve a better use of the WAN resources.

4.9.2.2 Performance Isolation for Network Slicing

In this section, we evaluate the performance of the slice isolation approach in terms of both delay and bandwidth guarantees in Section 3.2.1.2.2, considering the impacts not only on the data plane infrastructure but also on the extensions on the 5Growth architecture.

4.9.2.2.1 System Architecture

To enable the 5Growth platform utilizing performance isolation among network slices, our focus here is on the 5Gr-RL, interacting with the underlying data-plane infrastructure to provide different resources to network slices. In addition, two specific plugins within the 5Gr-RL are extended accordingly: (a) WIM plugin that enables control of the transport network and (b) VIM plugin for governing the available computing resources. In the following, we briefly elaborate some necessary extensions are done to formally evaluate the performance:

1. **5Gr-RL extensions**: The 5Gr-RL release is extended to support slice isolation. Specifically, a smart resource orchestration algorithm is triggered for each slice/NFV-NS, upon receiving a resource allocation request, mapping the abstracted requested resource to a QoS policy characterizing the network slice. For instance, the policy can include parameters like minimum/maximum bandwidth and maximum delay. Moreover, it sets up the corresponding alarms using the monitoring system. The 5Gr-RL southbound interface is also extended to

communicate parameters to the WIM plugin that is responsible for configuring transport infrastructures.

2. **5Gr WIM plugin:** Our extension to WIM plugin allows to handle the requests that aim to set up network slices with specific QoS characteristics and to enforce the performance isolation. Therefore, new parameters characterizing the slices are added, such as maximum/minimum bandwidth, maximum burst size, and Active Queue Management (AQM) parameters for delay guarantee, etc. After parameter extraction from the request, the WIM plugin interacts with the SDN controller to set up the slice and enforce QoS guarantees.
3. **5Gr VIM plugin**: The 5Gr-RL manages the VIM resources through the VIM plugin. Particularly, when Kubernetes is used as the VIM, the corresponding Kubernetes plugin is utilized for the 5Gr-RL. The VIM plugin first receives the 5Gr-RL's compute resource creation requests, translates it, and then sends it to Kubernetes to create pods inside the Kubernetes cluster. Note that the initial version of the Kubernetes plugin does not support VLAN-based networks, while different Container Network Interfaces (CNI) must be used by Kubernetes to support different network technologies. Hence, Multus CNI [89] is used to connect the pods to the transport network's external WIM. Multus is a CNI plugin for Kubernetes that enables attaching multiple network interfaces to pods, and it can create a VLAN-based network that connects a physical port to the pod. Thus, the VIM plugin is extended accordingly to provide the interface for managing the Multus CNI.

Moreover, to support the performance isolation over underlying data plane infrastructure, several extensions are realized over the control plane, such as ONOS SDN applications. First, to enforce traffic isolation, we employ the Virtual Private LAN Service (VPLS), deploying L2 multipoint-to-multipoint connections over the shared transport network. The VPLS native ONOS application is employed towards that end. Also, in order to be cross-compatible with data plane infrastructure beyond OvS, a re-engineering of the P4 data plane is executed to be applicable for P4 switches. Nevertheless, ONOS nowadays only provides traffic isolation across slices, without the capability of bandwidth and delay guarantees. Therefore, we introduce the *QoSlicing* as an application managing both meters and virtual queues across slices and exposing a unified interface towards the upper layers. Such application utilizes the developed *P4-pipeconf* application as the interpreter to interact with the underlying P4 pipeline, e.g., to obtain virtual queues' parameters. Finally, we can expose the performance-isolated network slices to verticals across different PoPs in the transport network, as the system architecture shown in Figure 77.

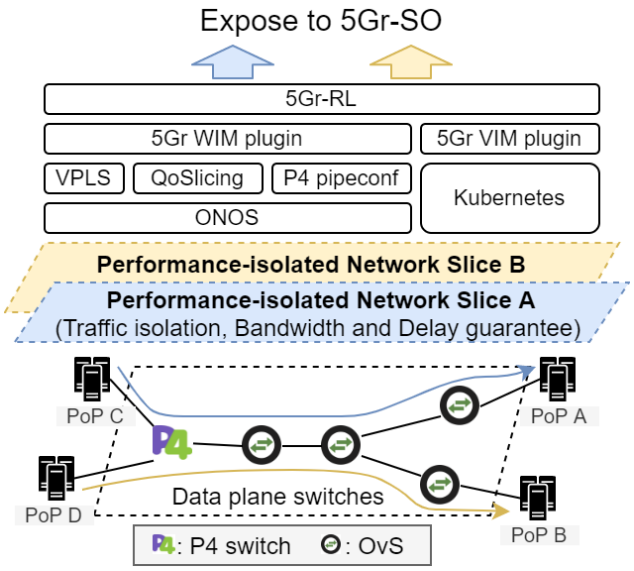


FIGURE 77: SYSTEM ARCHITECTURE OF PERFORMANCE ISOLATION INNOVATION

4.9.2.2.2 Emulation results

Before showing the evaluated performance, we first highlight four traffic metrics with regards to represent the performance of network slices. The first is the end-to-end latency by measuring the round-trip time between two end-points, and the second is the application-layer throughput by counting successfully received data payload. Moreover, to capture the latency in the data-plane infrastructure, we also measure two types of delay: virtual delay and real delay. The former is the latency encountered in the virtual queue, and the latter is the experienced delay of the actual physical queue in the pipeline. Note that these two types of delay show their importance under different conditions. A non-zero real delay represents that the physical queue is built up, while the virtual delay can limit throughput when link capacity is not saturated.

To demonstrate the system in Figure 77, one server hosting seven VMs is installed, with their interconnections and functionalities shown in Figure 78. There are four VMs acting as individual host to transmit/receive traffic via the correspondent PoP, while all data plane switches (OvS and P4 switch) are deployed in VM5 using the mininet [90] utility. Further, VM6 is used as the ONOS SDN controller with three abovementioned applications to control the underlying data plane switches, and the 5Gr-RL is on VM7 together with both VIM and WIM plugins.

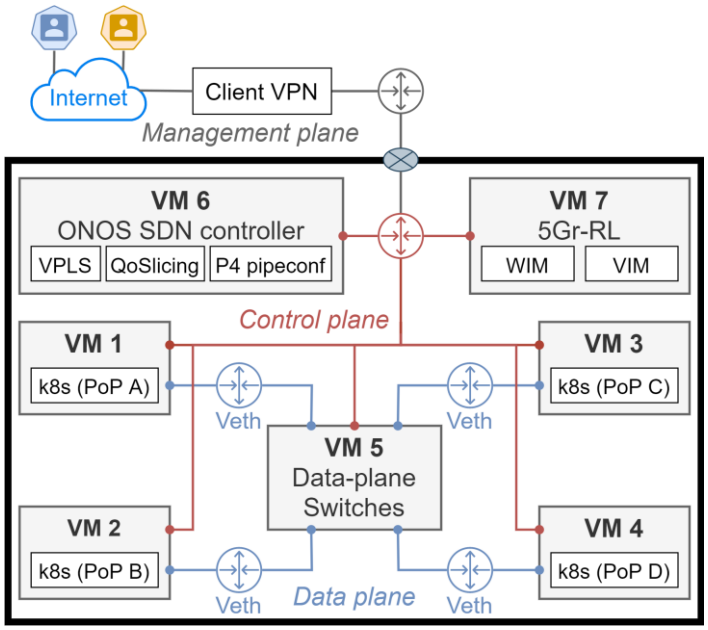


FIGURE 78: PROOF-OF-CONCEPT SETUP OF PERFORMANCE ISOLATION INNOVATION

Four presented phases in our demonstration are explained as follows, and the full video can be found in [94]:

1. **Phase 1**: We first set the link capacity to 72Mbps and instantiate one network slice A with a single 100 Mbps UDP traffic flowing from PoP C to PoP A. To show the vanilla behaviour of a network slice, neither bandwidth nor delay guarantee is applied. We can observe the result in Figure 79 with a very large end-to-end latency (more than 2 seconds) together with the saturated throughput. Moreover, a large real delay and almost zero virtual delay are observed, due to the link capacity full utilization and the physical queue is filled up.

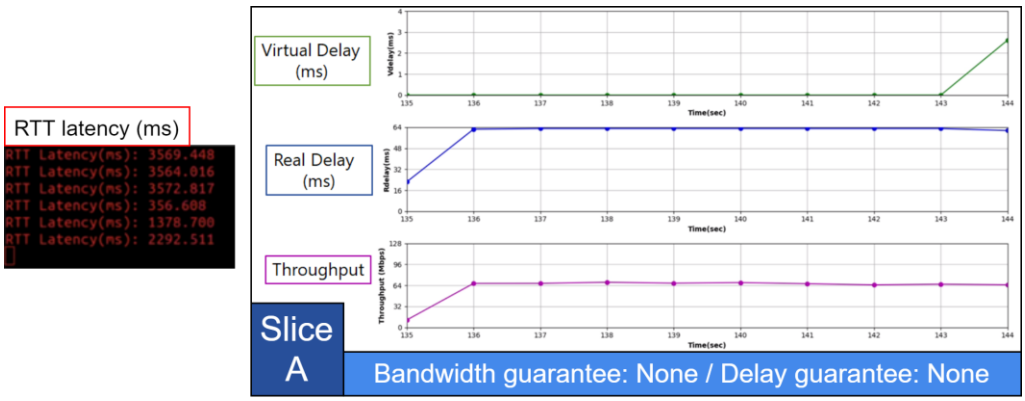


FIGURE 79: PHASE 1 RESULT OF PERFORMANCE ISOLATION INNOVATION

2. **Phase 2**: We then activate the performance isolation by configuring the slice A to guarantee 12Mbps throughput and 10ms delay. Since the guaranteed bandwidth is smaller than the link capacity, less than 1ms end-to-end latency is observed in Figure 80 together with no real delay, by reason of no physical queue being built up. In contrast, around 10ms virtual delay

is experienced using the virtual queue in the P4 pipeline and it can limit the throughput to the guaranteed 12Mbps.

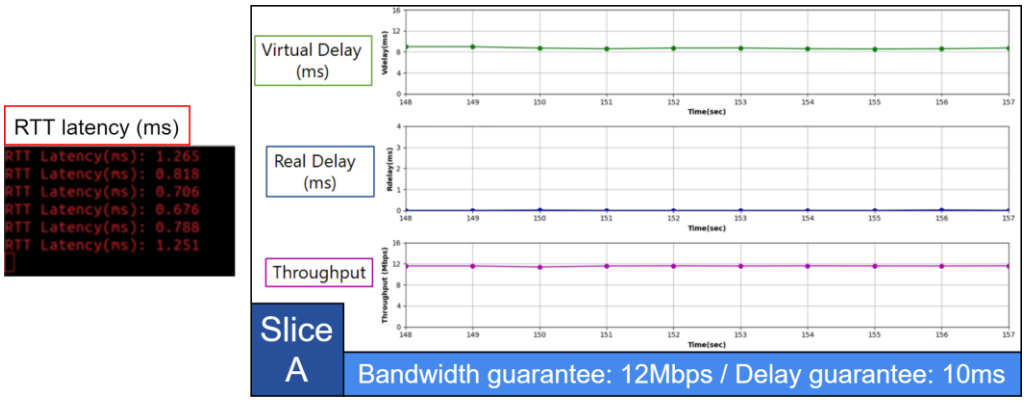


FIGURE 80: PHASE 2 RESULT OF PERFORMANCE ISOLATION INNOVATION

3. **Phase 3**: To see its effectiveness under different guarantees, we increase the guaranteed throughput to 120Mbps, exceeding the 72Mbps link capacity. We can see that both end-to-end latency and real delay are bounded to around 10ms with no virtual delay in Figure 81. This is due to the congestion at the real link, and thus virtual queue contributes no virtual delay.

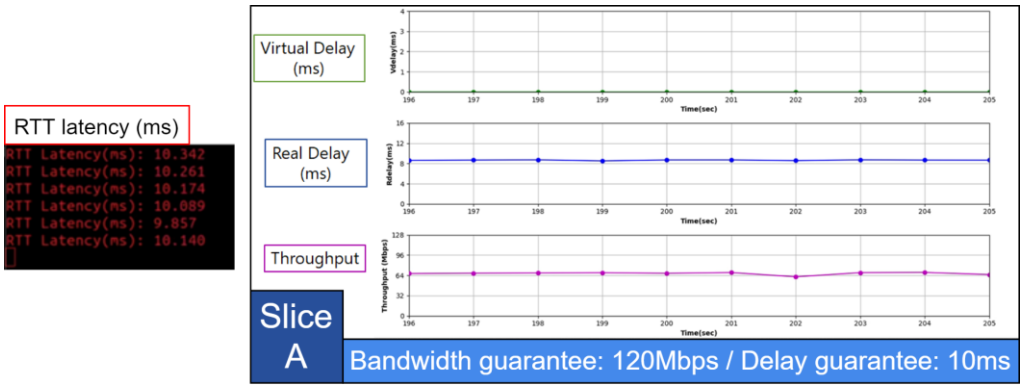


FIGURE 81: PHASE 3 RESULT OF PERFORMANCE ISOLATION INNOVATION

4. **Phase 4**: Another network slice B is brought up with the traffic flowing from PoP D to PoP B. Both slices are guaranteed with 48Mbps throughput, while slice B is with 5ms delay guarantee, being one half of the 10ms delay guarantee to slice A. The performance of slice A is then limited by the virtual queue due to its higher delay guarantee, while slice B is limited by the maximum link capacity and the real queue. As the results shown in Figure 82, slice A receives the guaranteed 48Mbps throughput with around 10ms virtual delay, while slice B gets the remaining link capacity with a larger real delay. We can notice that the slice with the most stringent delay bound determines who suffers from a lower throughput, while such slice prioritization can be modified with adjusted real queue priorities.

FIGURE 82: PHASE 4 RESULT OF PERFORMANCE ISOLATION INNOVATION

In the above evaluated performance, we can see that the 5Growth network slicing approach can retain traffic and performance isolation providing bandwidth and delay guarantees at the same time, based on our extensions over the 5Growth architecture and programmable data-plane infrastructure. Although this innovation is not planned to be deployed directly on-site with WP3 vertical trials due to the P4 switch matureness for industry-scale usage, we still plan to validate this network slicing innovation over multiple use cases from verticals in the lab environment.

4.9.3 Dynamic Profiling Mechanism

This section presents the evaluated outcomes of the proposed framework. The necessary data that were used for the evaluation were generated using the discrete-event network simulator (ns-3) All the experiments were conducted using a single PC unit equipped with Ubuntu 16.04, Intel® Core™ i7-6800K Processor 6/12 and 32 GB of DDR4 RAM. The simulated topology chosen is depicted in Figure 83. The data produced by ns-3, simulate 10800 real-life seconds (3 Hours), while the simulation's Estimated Time of Completion (ETC) is approximately 12 hours. The logging frequency of data is set to 1 second. The size of the area of experiments, in which the simulation was executed is equal to 400x200 m^2 with 10 equally sized sub-areas, in which a femtocell is placed in the middle. Additionally, there are 2 macrocells in the outer bounds of the area of experiments, which provide greater coverage if needed. There are 2 different mobility models implemented in the simulation. Each model has 2 parameters named velocity and path pattern. Both mobility models follow a random walk process for their path selection. The velocity of the low as well as the medium mobility

models is uniformly distributed in the ranges of $(0,0.2]\ m/s$ and $(0.2,1.2]\ m/s$ respectively. Table 26 summarizes the ns-3 parameters used in the simulated scenario.

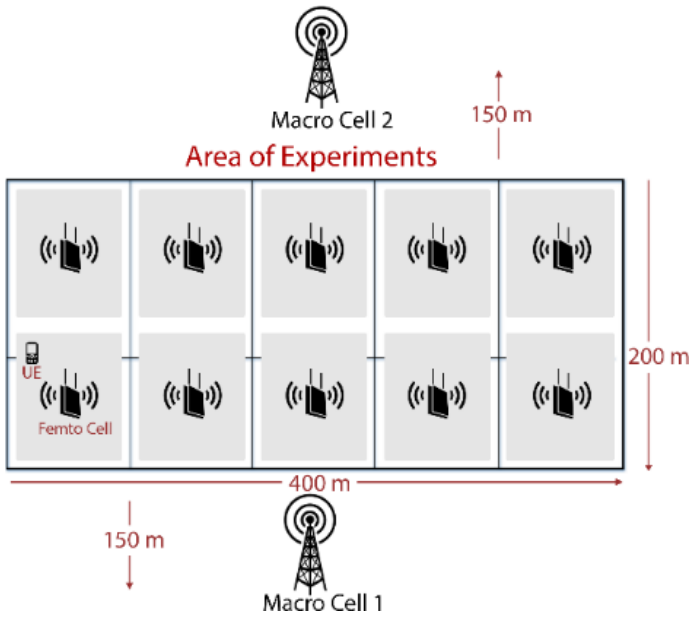


FIGURE 83: NS-3 VIRTUAL TOPOLOGY

TABLE 26: PARAMETERS USED IN THE NS-3 SIMULATED SCENARIO

Parameter Description	Default Value
# of Femtocells	10
# of Macrocells	2
Mobility Models	2 (Low, Medium)
UEs' Transmission Power	20 dBm
Femtocells' Transmission Power	20 dBm
Macrocells Transmission Power	35 dBm
Macrocells Downlink and Uplink Bandwidth	20 MHz
Femtocells Downlink and Uplink Bandwidth	20 MHz

In the current simulation scenario, a single UE consumes 3 different service types with diverse service traffic characteristics (Table 27).

TABLE 27: TRAFFIC MODELING SPECIFICATIONS OF THE SIMULATED SERVICES

	Transfer Protocol	Packet interval DL/UL	Packet size DL/UL
Service 1	UDP	2 / 2000 ms	600 / 12 bytes
Service 2	UDP	2 / 2 ms	1400 / 1400
Service 3	UDP	5 / 1000 ms	1100 / 12 bytes

During the simulation time, services are randomly assigned to the user and their duration is modelled following a normal distribution $X \rightarrow N(\mu,\sigma^2)\ where\ (\mu,\sigma^2) = \{(30,10),(45,10),(40,15)\}$ for each one of the abovementioned services, respectively. The selection concerning the mean value and variance aims at offering a rather small but distinctive variation in the duration for each one of the services.

4.9.3.1 Hierarchical Agglomerative Clustering, Dynamic Cluster Filtering and Profile Extraction evaluation

This section focuses on the evaluation of the HAC along with the implemented filtering mechanisms introduced in Section 3.2.1.3.3. Figure 84 presents the *Silhouette Scores* for the best parameters selected based on the results of the implemented grid search.

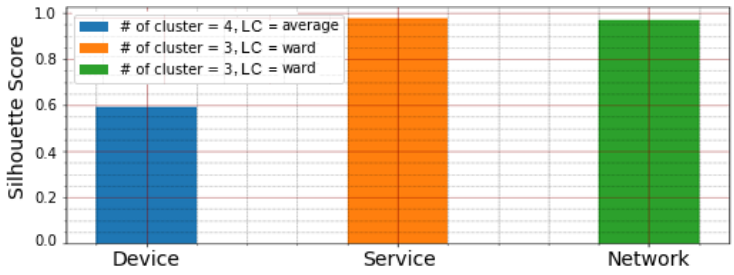


FIGURE 84: HAC PERFORMANCE (BEST SILHOUETTE SCORES) FOR THE 3ENTITIES: DEVICE, NETWORK, SERVICE

For the *Device* entity, the selected number of clusters parameter is set to 4, while the linkage criterion is set to *Average*. The *Silhouette Score* for this entity is nearly 0.6, indicating clustering results of low accuracy, since the data features of the *Device* are far less logically correlated and have no obvious repeating patterns. Regarding the *Service and Network* entities the number of clusters selected for both is set to 3, while the linkage criterion for both is set to *Ward*. For both the *Network* and *Service* entities, the *Silhouette Scores* are almost identical and close to 1 indicating that the entities have been separated into different clusters in an almost optimal manner. To this end, the rest of the evaluation is based on the aforementioned parameter configuration.

The next part that is presented in this section relates to the filtering mechanisms introduced in Section 3.2.1.3.3. More specifically, in Figure 85-a), 3 vertically stacked excerpt timelines, in which all the different clusters that were extracted from the HAC, are arranged based on their chronological occurrence. As it can be seen in the figure, the y-axis depicts the names of the different clusters and the x-axis depicts the duration of the timeline in seconds. In the time interval [4523, 4550] sec the *Service* and *Network* entities are rapidly switching clusters every second. These short-lasting clusters will be candidates for the outlier removal operation that follows. Figure 85-b) illustrates the behavioural cluster timelines as they are formed after the enforcement of the *One-Sec Removal Adaptation Operation*. Overall, the elimination process for the entire simulation, results in 3, 2 and 2 behavioural clusters for the entities respectively. The results of the filtering mechanism indicate that the output of the HAC could be improved (in terms of number of clusters) despite of the high *Silhouette Scores*. This can be explained by the fact that two out of the three simulated services have similar traffic load modelling, and -as a result- similar behaviour. As a result, the *One-Sec removal* operation is able to identify this similarity and remove the extra cluster for both *Service* and *Network* entities.

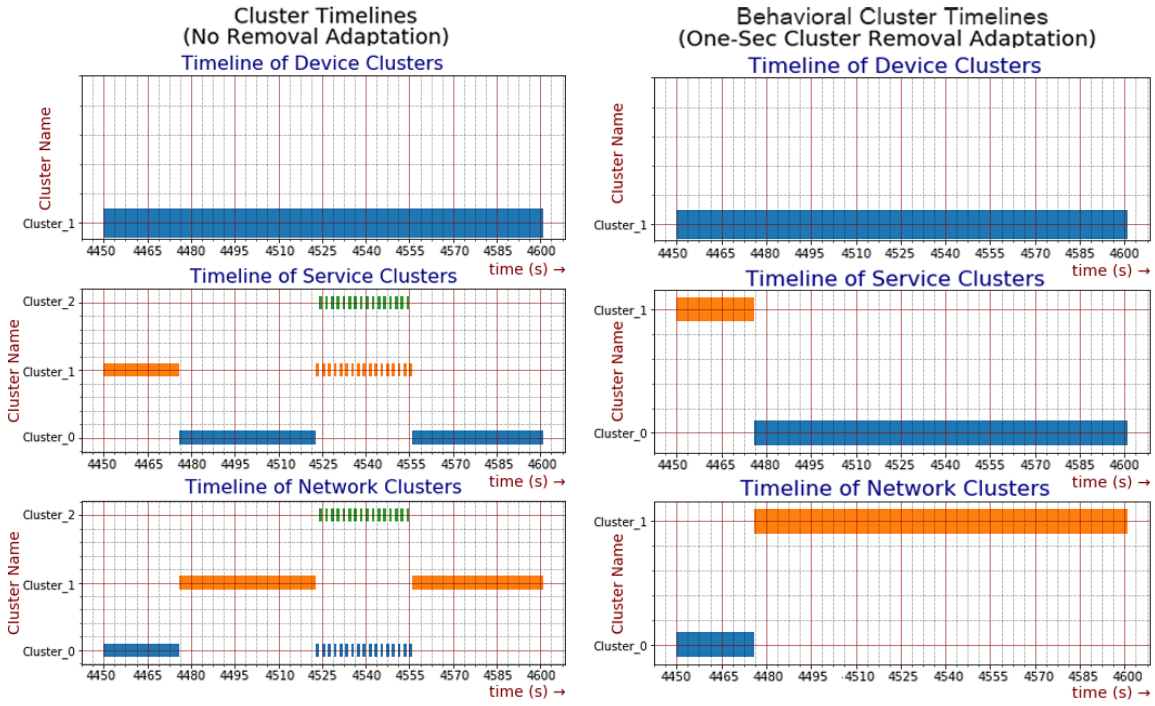


FIGURE 85: CLUSTER TIMELINES: (A) TIMELINES AS EXTRACTED FROM HAC, AND (B) TIMELINES AFTER THE ENFORCING OF THE ONE-SEC REMOVAL ADAPTATION OPERATION (BEHAVIOURAL CLUSTER TIMELINES)

Figure 86 presents the profile timelines that result after the application of the *Profile Mapping* along with the removal operations. It must be noted that some profiles appear to have no duration (P_0, P_1, P_4, P_8, P_10, P_12, P_14). This inability to depict the duration for some of the profiles, is explained by the fact that it is minuscule for such a large-scale timeline. Again, these short-lasting profiles will be candidates for the outlier removal operation. Finally, Figure 87 compares the different number of extracted profiles, under the above-mentioned filtering mechanisms. As it can be seen in the figure, there are 15 different profiles in total when no filtering mechanism is applied. After the application of the filtering operations, the different number of profiles are clearly reduced and equal to 10 and 6.

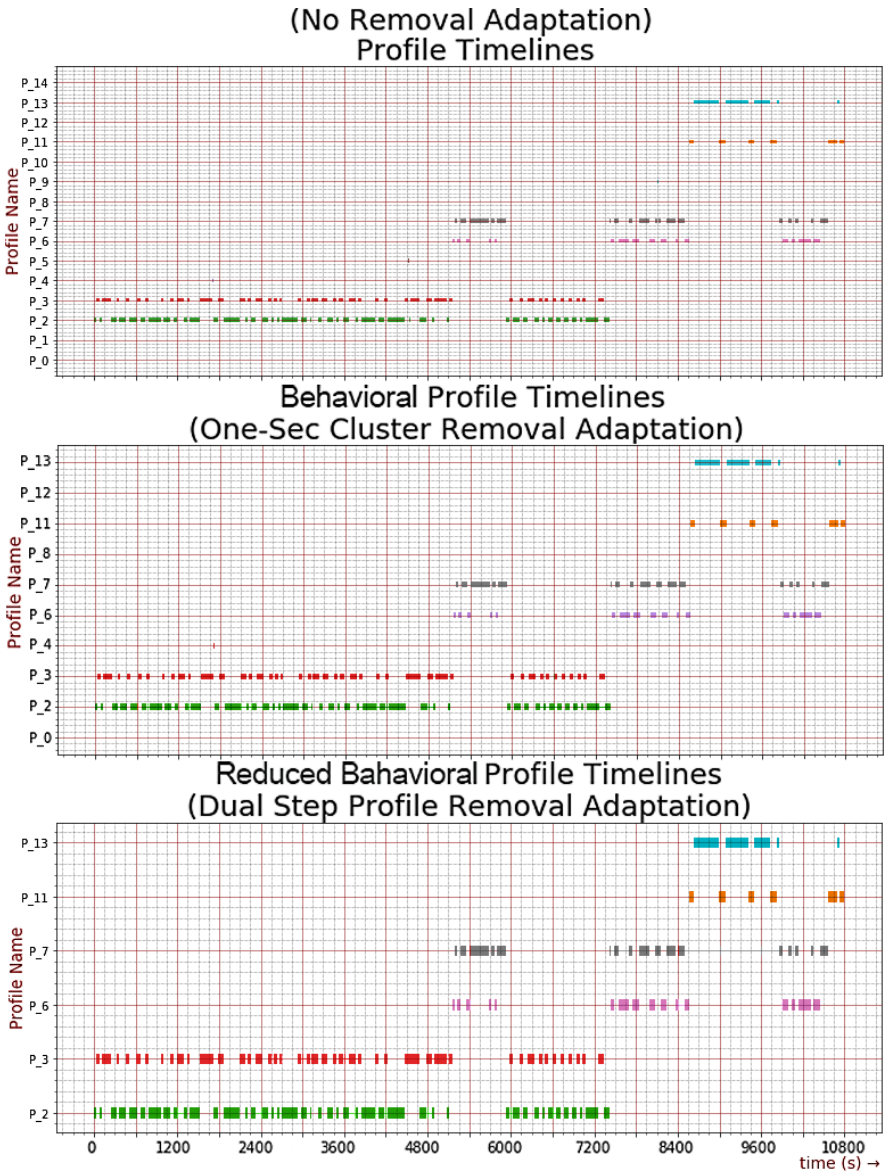


FIGURE 86: PROFILE TIMELINES: (TOP): WITH NO REMOVAL ADAPTATION (MIDDLE): WITH ONE-SEC REMOVAL ADAPTATION (BEHAVIOURAL PROFILES) (BOTTOM): WITH THE DUAL STEP REMOVAL ADAPTATION (REDUCED BEHAVIOURAL PROFILES)

FIGURE 87: DYNAMIC REMOVAL

4.9.3.2 Cluster and Profile Classification evaluation

This section focuses on the evaluation of the Weighted Voting Model. The weight of each vote is equal to 1 for the DTR and k-NN, while the voting weight for the case of RF and SVM models is doubled since they are the most robust algorithms among the rest. Table 28 depicts the performance of the aforementioned weighted model. The performance was measured using the accuracy and f1 score metrics.

TABLE 28: PERFORMANCE OF THE WEIGHTED VOTING MODEL

Entity Name	Accuracy	f1-Score
Device	96 %	92 %
Service	92.72 %	91.72 %
Network	93.68 %	89.68 %

4.9.3.3 Cluster and Profile Forecasting evaluation

In this last part of the evaluation, the results from the cluster and profile forecasting module are presented. After extensive experimentation regarding the training phase of the 3 LSTM models (one per entity), the batch size was set to 64 and the total number of epochs to 250. In order to prevent any degradation in the performance of the validation set and as a result overfitting, an early stopping function was applied. To increase the performance of the models the Stochastic Gradient Descent (SGD) with momentum was selected as an optimiser with learning rate $lr = 0.001$. All models used L2 (weight decay) regularisers of 10^{-6}. Finally, the time window selected for forecasting was set to 30 seconds using a 30 second prior time window of contextual information as an input. The split ratio for the dataset was set to 0.8. After segmenting the training and evaluation set into 30 second time windows, we result with 288x30 and 72x30 seconds of data, respectively.

Figure 88 presents the final behavioural profiles and the actual ones (ground truth) in order to illustrate the performance of the *Profile Extraction* mechanism. In this particular evaluation phase, a newly introduced profile, namely NP_0 is falsely forecasted according to the ground truth timeline. However, the aforementioned profile illustrates the capability of the module to forecast profiles, which were never observed before.

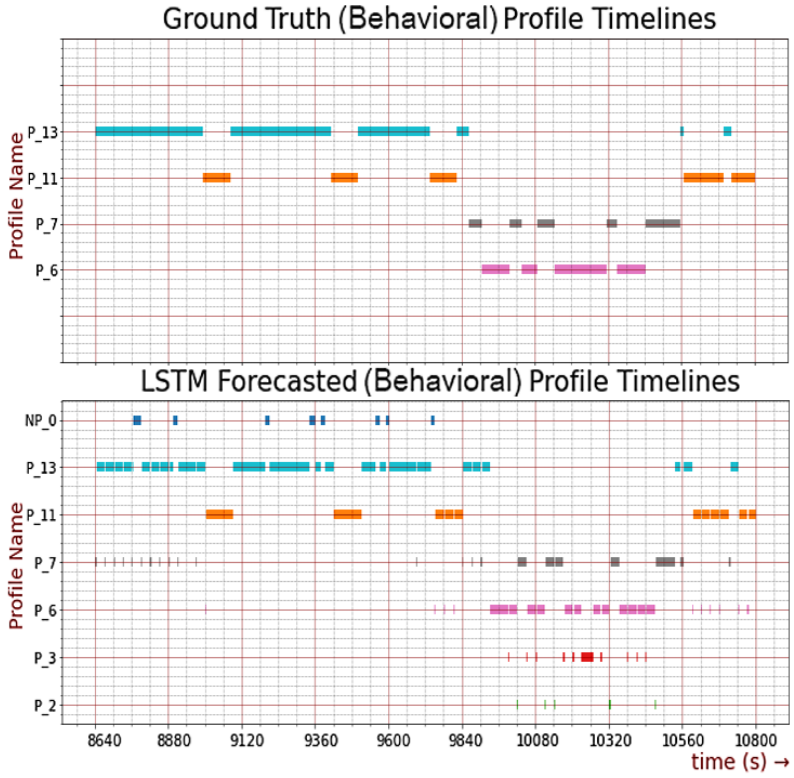


FIGURE 88: PROFILE TIMELINES: (A): FORECASTED (BEHAVIOURAL) PROFILES (B): GROUND TRUTH

FIGURE 89: PERFORMANCE OF CLUSTER & PROFILE FORECASTING MECHANISM

Figure 89 shows that the accuracy for the cluster forecasting is 88.29% for the device entity, 87.71% for the service entity and 82.59% for the network entity. Overall, the profile forecasting performance amounts to 71.25%.

A comprehensive methodology, which described step-by-step the modules and algorithms applied to the collected contextual network information towards profile extraction and forecasting, has been validated as described above. Novel definitions of entity clusters and profiles were introduced, described by a detailed data model, and a detailed evaluation study showed the effectiveness and viability of the proposed scheme.

4.10 Anomaly Detection

In this section, we evaluate the performance of the innovation introduced in Section 3.2.2 using the ns-3 discrete event simulator. The simulation duration is set to 600 seconds. The logging frequency

of the monitored data is set to 1 second. The area of experiments in which the simulation was executed is equal to 200x100 m^2, at the centre of which a femtocell is placed. The UEs follow a *Random-Walk* mobility model, with *medium* or *high* velocities, uniformly distributed in the range of $(0.2,1.2]\ m/s$ and $(1.2,2]\ m/s$ respectively. The simulated topology is depicted in Figure 90, while Table 29 summarizes the ns-3 parameters used in the scenario.

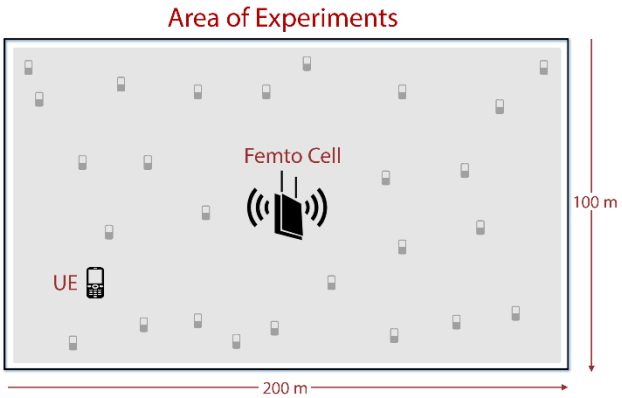


FIGURE 90: NS-3 VIRTUAL TOPOLOGY

TABLE 29: PARAMETERS USED IN THE NS-3 SIMULATED SCENARIO

Parameter Description	Default Value
# of normal behaviour UEs	1
# of traffic control UEs	30
# of Femtocells	1
Mobility states	2 (Medium, High)
UEs' Transmission power	20 dBm
Femtocells' transmission power	20 dBm
Femtocells downlink and uplink bandwidth	20 MHz

Overall, 31 UEs are simulated, one of which is considered as the *test UE* and operates under normal behaviour running two types of services. The traffic models of the *test UE* are in Table 30. The rest of UEs are used in order to increase/decrease the network load to simulate anomalous network conditions; the 30 loaded UEs operate in 2 modes *Idle* and *Overload*, in order to control the varying traffic load of the network, under the following configuration:

1. The UEs become active at $t = 60s$.
2. Every 30 seconds, each UE chooses a random mode.
3. For a specific 60 second time period $[420,480]$s, all UEs operate in a fixed Overload mode.
4. There are $3x30$ second time periods $[150,180], [270,300], [480,510]$ in which, all the UEs are in Idle mode.

The traffic models of the 2 aforementioned modes are presented in Table 31.

TABLE 30: SERVICE MODELING FOR THE NORMAL BEHAVIOUR UE (TEST UE)

Parameter	Service 1	Service 2
Transfer Protocol	TCP	UDP
Packet interval DL/UL	2/200 ms	1000/3 ms
Packet size DL/UL	500/50 bytes	12/1200 bytes

TABLE 31: TRAFFIC LOAD MODELING OF THE TWO MODES

Parameter	Idle Mode	Overload Mode
Packet Interval DL/UL	5000/5000 ms	1/1 ms
Packet Size DL/UL	12/12 bytes	1400/1400 bytes

The evaluation of the proposed scheme involves assigning a cluster to a new unseen and pre-processed (as in 1st phase of the Anomaly Detection Algorithmic Framework, Section 3.2.2) test dataset, using the already trained clustering algorithm to act as the ground-truth. The trained DNN model classifies each sample of the test set. The proposed scheme's accuracy is defined by calculating the percentage of correctly classified samples over the clusters assigned in the previous step. The classifier's accuracy varies based on the number of clusters determined at the beginning of the process. Figure 91 shows the accuracy score of the DNN classifier over the different number of clusters defined, where the y axis shows the percentage of correctly classified samples, averaging out in approximately 87.51% accuracy score. It is worth noting that as the number of clusters increases -- and hence, the algorithm's granularity -- a trade-off can be observed in the accuracy of the classification prediction. More specifically, for the 2-cluster configuration, the prediction accuracy reaches ~89,5%, while increasing to a 6-cluster configuration, the prediction scheme's performance decreases to ~82%, i.e., an ~8.3% loss. Thus, the configuration in terms of the number of clusters should be dependent on the specific use case and the granularity (i.e., number of levels) of the normal-anomalous behaviour range that the network administrator targets to identify.

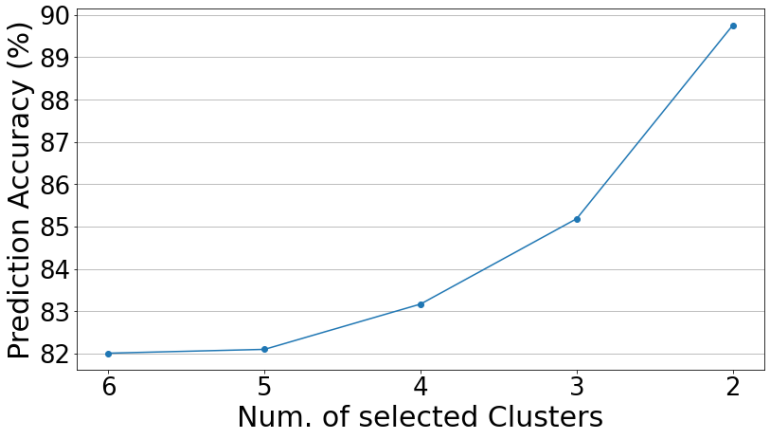


FIGURE 91: ANOMALY PREDICTION ACCURACY PERCENTAGE

4.11 Forecasting

This section presents a comparison among different forecasting algorithms that can be used by the 5Growth forecasting functional block (5Gr-FFB). To perform the performance comparison with real

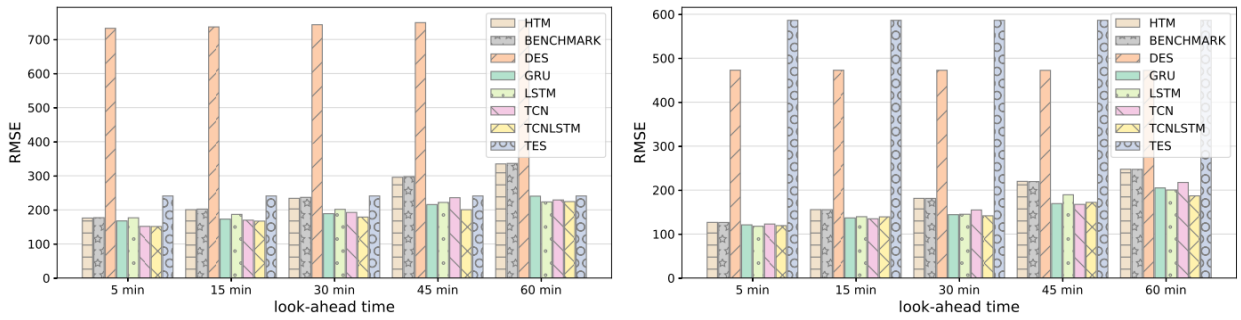
traces the time series representing the amount of resources requested by a Cellular Vehicle to Network (C-V2N) service is considered [101]. Specifically, it is assumed that the amount of resource requested by the C-V2N service is directly proportional to the amount of cars traveling along some city of Torino streets. However the comparison can be performed for any other application, provided the time series availability. The goal behind applying the forecasting is to implement forecast-based scaling of C-V2N services. Based on the forecast C-V2N service request, the scaling algorithm assigns enough resources to meet the vehicular service latency requirements. For this purpose, the relationship between the forecast flow of future cars and the service latency is based on a queuing mode where cars represent the clients while the available automotive service instances represent the servers. It is assumed that when cars enter in the crossing area of a street, they are requesting the service, as if mobile users' handover into another cell. The considered queuing model is an M/M/c model where *c* is the number of servers. Thus, the traffic flow for the next *n* minutes is forecast and the number of servers *c* is scaled up/down to meet the average delay computed through the queue model.

The techniques selected for performance comparison are assessed with respect to their forecasting accuracy and ability to adapt to changes in demand patterns. As such, several properties are taken into consideration in this analysis: (i) baseline performance; (ii) ability to forecast different periods in the future; (iii) capability to cope with changes in the patterns; and (iv) use of neighbouring information to improve forecasts.

The considered dataset for performing the forecasting technique evaluation contains information of Torino city traffic flows, with measures from more than 100 road probes reporting their location, traffic flow, and vehicles speed. The data is gathered by using S.I.MO.NE protocol [95]. Thus, probes' measurements are fetched every 5 minutes, and gathered in a common eXtensible Markup Language (XML) file.

Each forecasting technique is used to forecast the vehicles/hour traffic flow f_t seen in Corso Orbassano road probe 6. The forecasting techniques have been evaluated in two different scenarios with a 80% of training data, and a 20% of testing data, namely: non-COVID-19 scenario (training: 28th January - 28th February, testing: 29th February - 07th March); COVID-19 scenario (training: 06th February - 07th March; testing: 8th March - 15th March).

Figure 92 shows the accuracy in terms of Root Mean Square Error (RMSE) of the different forecasting techniques in predicting Corso Orbassano traffic as a function of the look-ahead time. The look-ahead time is the future time for which the data need to be forecast. Results illustrate how increasing the lookahead time forecast leads to an increasing RMSE in every possible training and dataset combinations, as it becomes more difficult to forecast the traffic further in the future.



(a) Offline training and non-COVID-19 scenario

(b) Offline training and COVID-19 scenario

FIGURE 92: ACCURACY OF SEVERAL FORECAST TECHNIQUES AS FUNCTION OF LOOK-AHEAD TIME

Figure 92(a) and Figure 92(b) show the RMSE values of offline training in non-COVID-19 and COVID-19 scenarios. The results presented in Figure 92(a) show that DES technique has the highest RMSE values, because the smooth (S_t) and the trend (T_t) values initially calculated during the training, are not updated in the testing phase. The other time-series technique (e.g., TES) mitigates such problem since its seasonality factor can capture better the trend. Figure 92(b) shows the RMSE values of the considered techniques in offline training with COVID-19 traffic. The considered scenario does not show any seasonality during 8th Mar - 15th Mar due to the COVID-19 lockdown. Thus, the obtained TES results exhibit the highest RMSE value compared to all other techniques.

The HTM (Hierarchical Temporal Memory) technique did not manage to outperform the sample-and-hold benchmark (i.e., $f_{t+1} = f_t$). For the rest of the techniques, the Neural Networks (NNs) solutions achieved the best performance for offline training. In the offline training, DES is not capable of capturing the trend, and the TES pitfalls in the COVID-19 scenario. Unlike DES and TES, the NN solutions can capture the evolving traffic trend thanks to the update of their hidden states (except the TCN). This explains why the NNs achieve lower RMSE when using offline forecasting (see Figure 92(a) and Figure 92(b)).

4.12 Moving Target Defense

The impact of forwarding plane performance is one of the most critical aspects, in which both latency and throughput are among the most used KPIs for end-application requirement specification. Therefore, in this section, we evaluate the forwarding plane network performance of the solution proposed in Section 3.3.1. We present in this document the most significant parameters for this evaluation, the complete set of experiments are available in the supporting article [96]. To exclude external variables' contribution in our results, such as NIC or cabling limitations, we will use a single large node (Virtual Machine (VM)) that containerizes the different networks using Linux network namespace. Each namespace will be terminated in a separate Open vSwitch bridge with its separate SDN controller instance. In effect, our network throughput is now limited just by the compute node's memory bandwidth and CPU power, revealing any performance bottlenecks that would otherwise not be noticeable when using NICs. Because our solution relies on the synchronization of clocks between the sending and receiving endpoints, we cannot evaluate that factor correctly when using

a single VM (i.e., both ends share the same clock). Therefore, we have also repeated the experiments using two separate VMs hosted in different compute nodes. The clock synchronization between the nodes was achieved using Network Time Protocol (NTP). We have validated that the solution behaves similarly to the single VM evaluation. As expected, the two VMs experiment was constrained by the NICs throughput and had added latency due to cable distance and network stack overheads to interact with the NIC. The added latency and jitter of that interconnecting network also influenced the mutation period's absolute fastest boundaries, but the findings remained similar. We must stress that, when independently reproducing the results, the Authenticator (SDN app) must be adjusted with the proper delay offsets caused by link latency and different computational capabilities that impact control-plane operations. In our case, both compute nodes had similar capabilities (same make and model, with Intel Xeon E5-2620 v4 CPUs) and were running the exact same implementation of the MTD mechanism. Therefore, we only had to account for an average symmetric link latency of approx. 0.46 ms (i.e., the offset used for authentication code calculation).

We measured link latency using a custom UDP client and server that can compute the one-way link latency through a timestamp (up to the microsecond) embedded into the packet payload. A new packet is generated every 0.1 ms. We have repeated **25000 × mutation_interval** runs for each mutation period to have statistically significant data at the timeslot change.

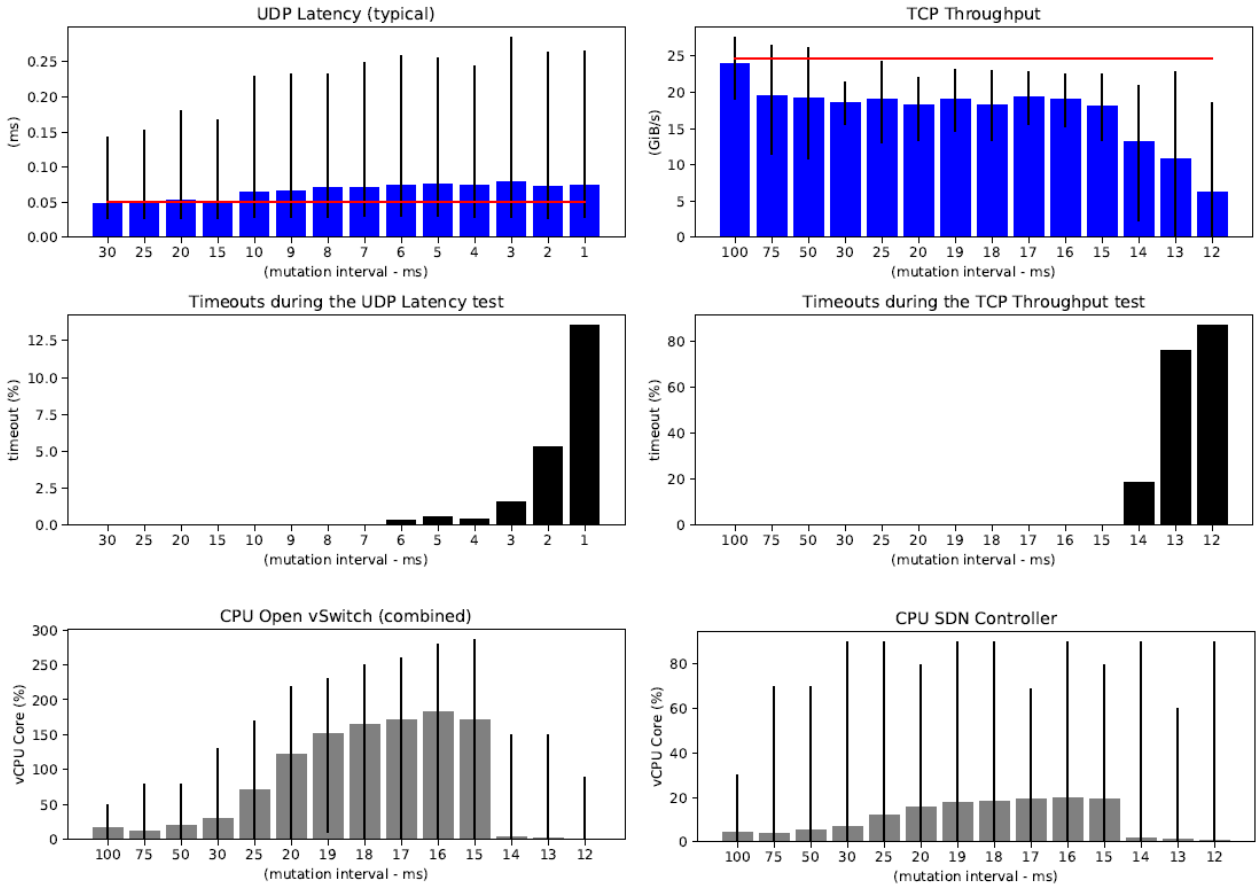


FIGURE 93: FORWARDING PLANE PERFORMANCE (BASELINE WITHOUT MTD IS IN RED)

As shown in the leftmost graphs of Figure 93, we have observed that the link latency closely resembles the measured latency without mutations up until the mutation period exceeds 15 ms. After that point, there is a slight increase whose total measured latency is < 0.1 ms on average. However, there was also a small increase in jitter, starting at 10 ms. Once the mutation period exceeded the 7 ms, the system started to drop packets, which were sent correctly to the controller and recorded in the threat assessment system as missed authentication codes. The 7 ms threshold is compatible with the SDN control overheads measured beforehand [96]. The 7 ms mutation period is the inflection point at which the deviation of the control-plane delays (5.44 ± 1.13 ms) must have started to become significant. After crossing this threshold, the mutation periods are no longer enforced correctly but rather as fast as the system can change the authentication codes (as verified by traffic logs). Because our PoC runs the same implementation on both ends within the same VM, the solution may keep working beyond the control-plane delays. The systematic errors will be very similar on both ends, achieving synchronization despite not enforcing that mutation period correctly. Although no longer keeping up with the mutation speed in these particular circumstances of a single node, the number of dropped packets still stays below 1% until 3 ms. After this point, it rises dramatically, starting at 2.25% in 3 ms, then 6.26% in 2 ms, and finally 14.40% in 1 ms. The experiments conducted using two nodes revealed that our straightforward implementation behaves similarly in UDP latency to the depiction in Figure 93 up-to-the identified threshold of the SDN control limits. After that, we get a more substantially rising packet loss, like the TCP throughput test's behaviour. We have opted not to include separate plots for the internode tests, as they are visually similar to Figure 93.

We have used iperf in TCP mode to measure the throughput of our solution. The client only uses one connection, and each test usually completes in about 10 seconds. We have performed at least 250 runs for each mutation period and set a timeout of 30 seconds for each test. We have also recorded the CPU usage during the test, focusing mainly on our virtual switch (that must mangle such a high number of packets) and the SDN controller (that must handle the PACKET_INs from mismatched authentication codes). The CPU load was measured using Python's psutil2 and a sampling interval of 100 ms to make the CPU bursts evident in otherwise lower averaged loads. Due to constraints imposed by our test setup, the recorded Open vSwitch CPU load pertains to both ends of the MTD connection (i.e., both sender and receiver). The SDN controller load refers to just one of the ends, and we have verified that both client and server had similar loads. The baseline refers to a pro-active SDN control approach in which the vSwitch gets a single catch-all rule that does the action NORMAL (i.e., a simple switch with no significant interaction with the controller). The throughput is very close to the average baseline when the mutation period is 100 ms (approx. 24.0 GiB/s vs. 24.65 GiB/s), falling within the error margin.

There were no connection timeouts or lost packets due to bad authentication codes in the 100 ms runs. However, the CPU load on both the vSwitch and the SDN controller was substantially higher than the baseline due to the frequent flow updates and higher demands of packet mangling versus plain pro-active forwarding. In the 75 ms to 15 ms mutation periods, the throughput lowers to approx. 18 to 19 GiB/s. The CPU load starts steeply rising as we change the packet mangling rules faster (see the OvS CPU load in Figure 93). While the throughput was quite good from 75 ms to 15

ms, and there were no connection timeouts during the test, we did observe a rising number of lost packets due to missing the right authentication code for that timeslot. Those lost packets were successfully recovered by the TCP session and are likely the throughput loss cause compared to the baseline. We will go more in-depth into these lost packets later in this section. The increasing number of PACKET_IN events from those missed authentication codes is also the likely cause for the rising CPU load in the SDN controller. The number of connection timeouts rises steeply after the 14 ms mutation period, and the solution becomes too unreliable after the 13 ms mutation period (over 75% connection timeouts). The CPU load dramatically falls as we start having timed-out connections. This reduction is because we have far fewer packets in the network while the TCP connection tries to recover. We must note that the throughput numbers only consider the runs in which a numerical value was acquired (i.e., without a timeout). Therefore, the throughput numbers for the mutation periods 14 ms to 12 ms must not be used to infer the total amount of data that went through the system in those runs.

Despite the straightforward implementation of the PoC, which did not feature any adaptive compensation for jitter in the system, the experimental results show that our solution can have little impact on the forwarding plane performance as long as the mutation period is correctly tuned. However, we do have non-negligible computational requirements for both the SDN controller and the virtual switch, which can rise very steeply as the mutation period accelerates. Nevertheless, because the vSwitch causes the most considerable CPU overhead and our solution employs SDN control, it may be possible to offload that CPU usage to a hardware OpenFlow switch.

Aiding in threat detection is one of our solution's main contributions, alongside making service enumeration and exploitation harder (i.e., stop unauthorized probing) to the attacker that does not hold the movement's secret. Because network jitter and other variances may cause delays that make an authentication code miss its targeted time slot, we need to evaluate our solution's effectiveness in separating these regularly occurring misses due to delays from the serious threats (i.e., identify the attackers trying to enumerate services). As the highly stressful TCP throughput tests have already shown, without considering the massive number of packets transacted in the network, the raw number of flagged events for further threat analysis even without adversarial action was substantial (approx. 841 thousand events throughout those tests). We will start by looking into the experimentally more controllable UDP traffic to analyse those authentication codes misses and determine why those misses happened despite having no adversarial action. Then, we will introduce a method to eliminate the false positives. Finally, we will reassess the capabilities to stop and detect adversarial actions in the light of the false-positives elimination.

We found a noticeable latency increase in the forwarding plane performance section when going from 15 ms to 10 ms. After repeating the experiment, we took a more in-depth look into the missed authentication codes in that threshold's vicinity. The results show there are significant differences between the before and after that threshold. Before crossing the 15 ms mark, we have very few authentication codes misses – only four were recorded. We could determine the missed timeslot in three of them, but the other miss was completely extemporaneous. While we have very few misses in the [30, 15] ms mutation interval, a miss will be due to a considerable delay (more than 100

timeslots). In contrast, when the mutation period yields an increased forwarding-plane latency over the baseline ([14, 10] ms), there will be more missed authentication codes. However, the vast majority is just a few timeslots apart (averaging nearly two slots). This finding is crucial to eliminate the false positives, especially when no additional monitoring data is available from the platform, such as in our straightforward PoC.

Our solution allows setting different grace periods (i.e., valid authentication codes) for stopping suspicious behaviour (i.e., forwarding plane rules) than when alerting suspicious behaviour (i.e., threat decision alarm generation). That is, we can reduce the number of alarms that need more in-depth inspection while still enforcing stricter tolerances on the forwarding plane by merely setting different grace periods in the respective components. Our configuration space has 65535 values possible (the number of service ports). For example, suppose our threat decision system validates the nearby [-655, +655] timeslots before generating an alarm. In that case, we will cover all previously identified events while still only having an approx. 2% theoretical probability of creating a false-negative. There is just one false positive that survives this elimination process, in over 7.5 million messages considered for the experiment. The threat assessment system could quickly eliminate such a false positive using additional anti-malware functions over the suspected packet/flow. In the worst case, the system administrators should handle a single false alarm easily.

In sum, the system's security relies on an adjustable trade-off between false negatives and false positives, which is dictated by the available resources, additional anti-malware functions, and monitoring data. We have shown that our solution allows us to eliminate almost all false reports generated when there is no adversarial action. We will now move to the other goal, demonstrating that it can detect adversarial action (probing attempts) against the network.

We have placed an attacker into the protected network, which somehow already identified the NF address that holds the sensitive NS (e.g., through the same configuration leakage that allowed him into that network). The attacker generates a single probing attempt, at a random time, in a bid to identify the listening port of that sensitive service while remaining undetected. We measured the number of times the attacker avoids detection through a Monte Carlo experiment. The attacker uses the best method to guess the service port, a randomly generated port. We repeated the experiment 100 million times to get statistically significant data. The threat detection happens in two stages. The first is in the forwarding plane, where the rules' enforcement only allows a tolerance of [-1, +1] timeslots, blocking and reporting any mismatching packets to the threat assessment system. The second stage is after the blocking event in the forwarding plane; the threat detection system will assess that event and determine if an alarm needs to be generated or if that report is due to regularly occurring conditions (e.g., CPU load or network jitter). The grace period is the same as used to filter the false positives, [-655, +655] timeslots. The attacker may avoid detection in two different situations. The first and most severe, in which the attacker matches one of the valid authentication codes in the forwarding plane and hits the real service, has a measured probability of 0.0045 ± 0.0005%. The second and less severe one is when the probing attempt was stopped in the forwarding plane (i.e., no useful outcome to the attacker). However, the threat assessment system produced a false-negative from that reported packet, therefore not generating an alarm. The probability of

having a false negative was measured to be 1.9768 ± 0.0099%. Therefore, when both cases are considered together, the measured probability of that single probe not generating an alarm is 1.9813 ± 0.0099%. We must note that if the attacker attempts to do multiple probes, then our chances of detection improve substantially, which curbs the asymmetric relationship the attackers had over defenders.

The results demonstrate our solution's effectiveness, as an extra layer of security, in stopping attacks against exposed network services and the soundness of the options made from the previous section's experimental data to remove the false positives. Furthermore, the PoC in the evaluation was a straightforward approach that does not leverage newer technologies (such as AI/ML) to improve delay prediction. The PoC also did not have access to live monitoring data accurately depicting the network's and compute nodes' statuses. Despite that, the experimental results already show that the proposed 5Gr-MTD solution is sound.

4.13 5Growth Platform Deployment CI/CD

As described in Section 3.3.2, CI/CD includes the integration and deployment pipelines and platform containerization. Please refer to that section for more innovation details. The goal of this section is to evaluate CI/CD and containerization benefits in reducing the time and resources required by 5Growth platform deployment. The evaluation was conducted to compare two scenarios. The first one is a 5Growth platform deployment using a containerized 5Growth platform and CI/CD. The second one is a 5Growth platform deployment on a virtual machine without CI/CD.

Both scenarios consider the following 5Growth platform components and infrastructure deployment and don't include 5Gr-AIMLP and 5Gr-FBB components:

- VM instantiation
- 5Gr-VS;
- 5Gr-SO;
- 5Gr-RL;
- 5Gr-VoMS.

The overall deployment flow of scenario 1 can be found in Figure 94. The 5Growth platform deployment using 5Growth CI/CD considers a platform deployment on a container capable environment (i.e., Kubernetes). For deployment, CD uses artefacts prepared on the CI stage. The CI flow and architecture are described in detail in Section 3.3.2. These artefacts include prebuilt images for every 5Growth component, configuration for 5Growth components and standard images, required by a deployment, and predefined Kubernetes manifests to put all things together.

Based on the predefined Kubernetes manifests, the overall deployment pipeline will interact with the infrastructure orchestrator (Kubernetes in this case) and passes instructions to it. Specifically, according to these instructions, the orchestrator will:

- Create entities (e.g., pods, services, configmaps, secrets);
- Establish network connections (e.g., IPs, DNS names, route external connections);

- Handle configuration options (e.g., mount cofigmap and secret entities); and
- Download images.

Due to the container's nature, the process of building and installation are not time-consuming, and the prebuilt images from the CI stage can promptly start their work.

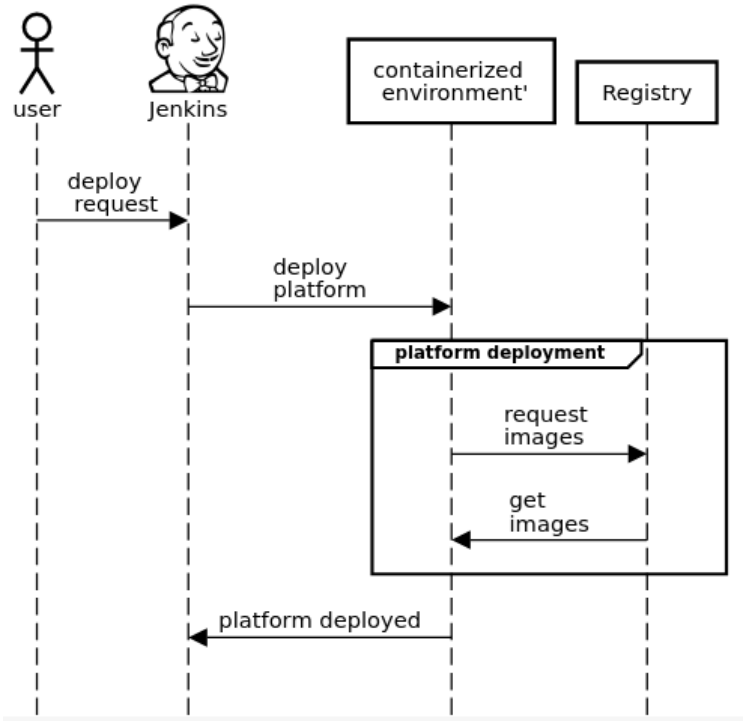


FIGURE 94: 5GROWTH CD PLATFORM DEPLOMENT WORKFLOW

On the other hand, the overall deployment flow of scenario 2 can be found in Figure 95. Such 5Growth platform deployment without 5Growth CI/CD (scenario 2) considers platform deployment on VM in a non-containerized environment. In particular, it includes the following actions:

- Instantiate VM;
- Install prerequisites;
- Install platform required software (databases, message queue (MQ), etc.);
- Sources download;
- Build platform components binaries;
- Configure platform; and
- Run platform.

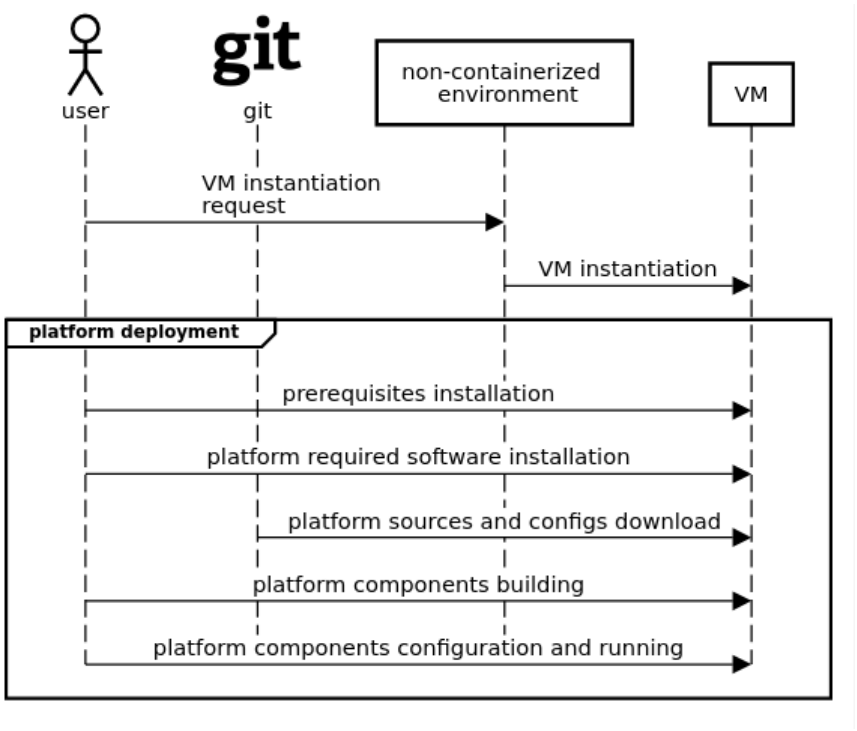


FIGURE 95: 5GROWTH PLATFORM DEPLOYMENT WITHOUT CD

The measured performance results of these two scenarios can be found in Table 32 and in Figure 96 in this case, for a platform consisting of 5Gr-VS, 5Gr-SO, 5Gr-RL, and 5Gr-VoMS. Stage time consumption representation is presented in Table 32. It is worth noting that the container capable environment (Kubernetes), VM orchestrator (OpenStack) and built image storage (local docker registry) all are parts of the CI/CD infrastructure deployed at 5TONIC lab [98]. We can see from the result that Scenario 1 is significantly faster because containers do not require the instantiation, building, and platform required software installation stages. We can note that the building stage requires significant resources for compilation, and thus the absence of this stage in scenario 1 can decrease resource consumption. Within scenario 1, we can see that the most time-demanding stage is the images download stage and the overall deployment only take tens of seconds.

TABLE 32: PER-STAGE TIME CONSUMPTION OF TWO SCENARIOS

Stages	Scenario 1: CD deployment time (sec)	Scenario 2: Non-CD deployment time (sec)
VM instantiation	-	2.59
5Gr-VS building	-	14.7
5Gr-SO building	-	6.5
5Gr-RL building	-	7.6
5Gr-VoMS building	-	13.65
5Gr-VS run	1.3	1.1
5Gr-SO run	1.3	1.4
5Gr-RL run	7.3	7.2
5Gr-VoMS run	1.8	1.7
Total	11.7	56.44

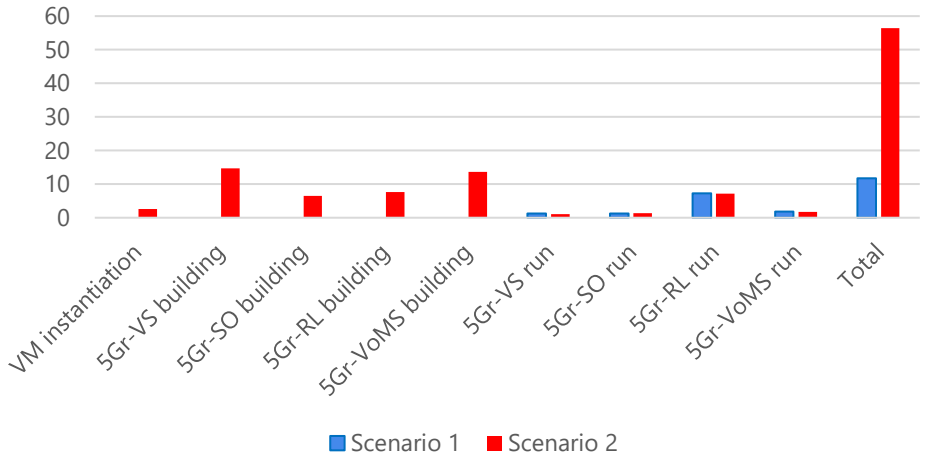


FIGURE 96: DEPLOYMENT TIME COMPARISON

5 Conclusions

Being the most research-oriented WP of the project, the goal of WP2 was to analyse the gaps detected in the architecture that was taken as a baseline, designed in the framework of a 5GPPP Phase 2 project (5G-TRANSFORMER), and move it to the next level, hence providing a platform closer to exploitation together with WP3 and WP4. As such, many of the architectural building blocks have been substantially extended (5Gr-VS, 5Gr-SO, 5Gr-RL, 5G-VoMS) and new blocks have been integrated to add a new set of capabilities (5Gr-AIML, 5Gr-FFB). All the added functionality has been selected after carrying out a detailed analysis (together with WP1 and WP3) of the needs of the vertical use cases of interest in this project for automating the service instantiation (e.g., reducing deployment time through automation) and adding intelligence adapting to operational in real-time, hence contributing to reduce OPEX and to make the network mode dependable. In this way, all the WP2 innovations reported in Section 3 cover the gaps as described in Table 33.

TABLE 33: MAPPING BETWEEN PLATFORM REQUIREMENTS (GAPS) AND SELECTED INNOVATIONS

Feature (gap)	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10	I11	I12
Enhanced VS network slice sharing				X	X							
VS arbitration at runtime		X	X	X	X			X				
VS layer federation						X						
VS dynamic service composition				X		X		X				
SO automatic network service management		X	X	X	X			X	X	X		X
SO self-adaptation actions		X		X	X	X		X	X	X		
SO dynamic monitoring orchestration		X	X	X					X			
SO geo-location dependent federation				X		X						
RL PNF integration	X						X					

Furthermore, the potential application in pilots of the new features developed was also discussed and offered as input to WP3/WP4. Since there are multiple features, there is a variety of potential application ranging from those that are trsnversal and of application to most pilots (e.g., those related with self-adaptation actions) and others that have more interest for a pilots of a certain type (e.g., VS layer federation). A complete discussion is rpesented in Section 3.4.

In addition to filling up the above functional gaps, these innovations were also designed to make an as efficient as possible use of resources, hence acting as enablers of dense deployments, such as those expected in future networks and future vertical scenarios. Multiple simulations and proof-of-concepts have validated the innovations presented in this document. Therefore, they are ready to be fully exploited [103] during the project lifetime and also after it in the form of, for instance, products,

services, patents, open source code, and in other research projects that set the foundations for 6G, since many of the architectural concepts analysed here can become valuable inputs in this process. Among these, it is worth highlighting the following results:

- 5Growth can instantiate end-to-end network slices well within minutes (see Section 4.1);
- 5Growth can swiftly manage AI/ML models as a service to other components of the stack to aid in network service scaling operations efficiently (see Section 4.3);
- 5Growth can inter-connect with other domains, such as 5G-VINNI (see Section 4.5);
- 5Growth can attain substantial OPEX/CAPEX savings in virtualized RANs (see Section 4.8);
- 5Growth can autonomously scale services and provide resource arbitration across slices efficiently (see Section 4.9.1);
- 5Growth can attain performance isolation across network slices, and provide slices with performance guarantees in network bandwidth and delay (see Section 4.9.2);
- 5Growth provides novel security mechanisms that can protect all 5Growth interfaces (see Section 4.12).

These results, among others properly reported in Section 4, validate that WP2 has contributed to meet the 5GPPP KPIs associated with the following tehnical objectives of the project:

Reduce network management OPEX by at least 20% compared to today.

We have shown in our results in Section 4.8 and 4.9.1.4 that an appropriate orchestration of network slices, which is achieved thanks to features from Innovations 1, 2, 3, 4, 5, 7 and 8, we can meet (and exceed, during low-load regimes) this target.

Create a secure, reliable and dependable Internet with a "zero perceived" downtime for services provision.

The SLA-preserving scaling algorithms and performance isolation techniques from Innovation 8, our anomaly detection algorithms (Innovation 9), and the security features from Innovation 11, assessed in Sections 4.9.1, 4.9.2.2, 4.10, and 4.12, validated this result at length.

Reduce the average service creation time cycle from 90 hours to 90 minutes.

The results we have shown in Sections 4.1, 4.4, 4.6, 4.7 do validate the ability of 5Growth to create a service, and preserve its SLA, in less than a few minutes. Features from Innovation 1, 3, 4, and 6 contributed to this achievement.

Facilitate very dense deployments of wireless communication links to connect over 7 trillion wireless devices serving over 7 billion people.

The focus on resource optimization in many innovations such as Innovation 1, 4, 5, 7 and 8 contribute to this KPI by maximizing the efficiency of the available resources, which makes networks amenable to denser scenarios. In addition, our security, anomaly detection, and forecasting mechanisms (Innovations 9, 10, and 11), shall help to preserve reliable services and quick reaction times upon changing events, common in such dense scenarios. The results presented in Sections 4.6, 4.7, 4.8, 4.9, 4.10, 4.11 and 4.12 are evidence of this contribution.

6 References

[1] 5Growth "Initial Design of 5G End-to-End Service Platform." Deliverable D2.1, December 2019

[2] 5Growth Github repository. Available at: https://github.com/5growth

[3] 5Growth "Initial implementation of 5G End-to-End Service Platform" Deliverable D2.2, May 2020

[4] 3GPP, "3GPP TS 28.531 Technical Specification Group Services and System Aspects; Management and orchestration; Provisioning; (Release 16)" March 2020.

[5] ETSI GS NFV-IFA 013 "Network Functions Virtualisation (NFV); Management and Orchestration; Os-Ma-Nfvo reference point - Interface and Information Model Specification", 2016.

[6] ETSI GS NFV-IFA 014 "Network Functions Virtualisation (NFV); Management and Orchestration; Network Service Templates Specification", 2016.

[7] ETSI GR NFV-IFA 028 "Network Functions Virtualisation (NFV) Release 3; Management and Orchestration; Report on architecture options to support multiple administrative domains", 2018

[8] ETSI GR NFV-IFA 005 "ETSI GS NFV-IFA 005 V3.1.1 (2018-08) Network Functions Virtualisation (NFV) Release 3; Management and Orchestration; Or-Vi reference point - Interface and Information Model Specification", 2018

[9] 5Growth "Final implementation of 5G End-to-End Service Platform." Deliverable D2.4, May 2021

[10] Apache Hadoop library. Available at: https://hadoop.apache.org/

[11] Apache Spark analytics engine. Available at: https://spark.apache.org/

[12] Big-DL distributed deep learning library for Apache Spark. Available at: https://bigdl-project.github.io/

[13] Ray library. Available at: https://ray.io/

[14] 5G-TRANSFORMER, "Final design and implementation report on service orchestration, federation, and monitoring platform", Deliverable 5G-TRANSFORMER D4.3, May 2019.

[15] J. Baranda, et. al., "NFV Service Federation: enabling Multi-Provider eHealth Emergency Services", in Proc. of IEEE INFOCOM 2020, July 2020

[16] 3GPP, «3GPP TS 28.531 Technical Specification Group Services and System Aspects; Management and orchestration; Provisioning; (Release 16),» March 2020.

[17] 5G-TRANSFORMER, "Final design and implementation report on service orchestration, federation, and monitoring platform," Deliverable 4.3, May 2019.

[18] Official Prometheus Pushgateway. Available at: https://github.com/prometheus/pushgateway

[19] Papagianni, Chrysa, et al. "5Growth: AI-driven 5G for Automation in Vertical Industries." European Conference on Networks and Communications (EuCNC), 2020.

[20] ETSI GS NFV-IFA 013, "Network Functions Virtualisation (NFV) Release 3; Management and Orchestration; Os-Ma-Nfvo reference point – Interface and Information Model Specification" v3.2.1, April 2019.

[21] ETSI GS NFV-IFA 014, "Network Functions Virtualisation (NFV) Release 3; Management and Orchestration; Network Service Templates Specification" v3.2.1, April 2019.

[22] 3GPP TS 28.530, "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Management and orchestration; Concepts, use cases and requirements

[23] 5G-TRANSFORMER, "Final design and implementation report on the MTP: Design," Deliverable D2.3, May 2019.

[24] 3GPP TS 28.541, "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Management and orchestration; 5G Network Resource Model (NRM); Stage 2 and stage 3 (Release 16)", v16.2.0, September 2019

[25] 3GPP TS 23.501, V16.0.2, "System architecture for the 5G System (5GS)", April 2019.
[26] 3GPP TS 23.502, V16.0.2, "Procedures for the 5G System (5GS)", April 2019.
[27] GSMA, "NG.116 Generic Network Slice Template v4.0". Available at: https://www.gsma.com/newsroom/wp-content/uploads//NG.116-v4.0-1.pdf, November 2020
[28] 3GPP TS 28.622, V16.7.0, "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Telecommunication management; Generic Network Resource Model (NRM); Integration Reference Point (IRP); Information Service (IS) (Release 16)", March 2021
[29] ETSI GS NFVI-IFA 013, "Network Functions Virtualisation (NFV) Release 3; Os-Ma-nfvo reference point - Interface and Information Model Specification", v3.4.1, June 2020.
[30] ETSI GR NFV-IFA 024, "Network Functions Virtualisation (NFV) Release 3; Information Modeling; Report on External Touchpoints related to NFV Information Model", v3.2.1, April 2019
[31] J. Baranda, J. Mangues, R. Martínez, L. Vettori, K. Antevski, C. J. Bernardos, X. Li, Realising the Network Service Federation vision , IEEE Vehicular Technology Magazine, Future Networks Initiative Special Issue on 5G Technologies and Applications, June 2020.
[32] ETSI GR NFV-EVE 012, "Network Functions Virtualisation (NFV) Release 3; Evolution and Ecosystem; Report on Network Slicing Support with ETSI NFV Architecture Framework" v3.1.1, December 2017.
[33] Wireguard VPN tunnel. Available at: https://www.wireguard.com/
[34] ETSI GR NFV-IFA 028, "Report on architecture options to supportmultiple administrative domains", Jan 2018.
[35] S. D. Jap, "The impact of online reverse auction design on buyer–supplier relationships," Journal of Marketing, vol. 71, no. 1, 2007.
[36] 3GPP, «3GPP TS 28.801 Telecommunication management; Study on management and orchestration of network slicing for next generation network; (Release 15),» April 2017.
[37] ETSI GS NFV-IFA 011, "Network Functions Virtualisation (NFV) Release 3; Management and Orchestration; VNF Descriptor and Packaging Specification" v3.2.1, April 2019.
[38] Ayala-Romero, J.A., Garcia-Saavedra, A., Gramaglia, M., Costa-Perez, X., Banchs, A. and Alcaraz, J.J. vrAIn: Deep Learning based Orchestration for Computing and Radio Resources in vRANs. *IEEE Transactions on Mobile Computing*.
[39] Demonstration of an experimental Proof-of-Concept of vrAIn, a deep-learning based orchestrator for virtualized RANs. Available at: https://youtu.be/1I8mcnHQcW8
[40] O-RAN Working Group 2: AI/ML workflow description and requirements, Technical Report "O-RAN.WG2.AIML-v01.01"
[41] J. Baranda et al., "On the integration of AI/ML-based scaling operations in the 5Growth platform," i IEEE Conf. on Network Function Virtualization and Software Defined Networks (NFV-SDN).2020.
[42] J. Mangues et al, "5G-TRANSFORMER Service Orchestrator: Design Implementation and Evaluation", in Procs. of the the European Conference on Networks and Communications (EUCNC'19), 18-21 June 2019, Valencia (Spain).
[43] L. Magoula, S. Barmpounakis, I. Stavrakakis, N. Alonistioti, "A genetic algorithm approach for service function chain placement in 5G and beyond, virtualized edge networks," Computer Networks, 2021.
[44] J. Sun, F. Liu, M. Ahmed and Y. Li, "Efficient Virtual Network Function Placement for Poisson Arrived Traffic," ICC 2019 - 2019 IEEE International Conference on Communications (ICC), Shanghai, China, 2019.

[45] Thomas, B. (1996). Evolutionary algorithms in theory and practice.
[46] Hedar, Abdel-Rahman & Ong, Bun & Fukushima, Masao. (2007). Genetic algorithms with automatic accelerated termination.
[47] Kunche P., Reddy K.V.V.S. "Heuristic and Meta-Heuristic Optimization. In: Metaheuristic Applications to Speech Enhancement." SpringerBriefs in Electrical and Computer Engineering. Springer, Cham, 2016.
[48] T. Subramanya and R. Riggio, "Machine Learning-Driven Scaling and Placement of Virtual Network Functions at the Network Edges," 2019 IEEE Conference on Network Softwarization (NetSoft), Paris, France, 2019.
[49] H. Krishna, N. L. M. van Adrichem, and F. A. Kuipers, "Providing bandwidth guarantees with OpenFlow," in 2016 Symposium on Communications and Vehicular Technologies (SCVT), 2016.
[50] M. S. Al Breiki, S. Zhou, and Y. R. Luo, "A Meter Band Rate Mechanism to Improve the Native QoS Capability of OpenFlow and OpenDaylight," in 2019 International Conference on Advanced Communication Technologies and Networking (CommNet), 2019.
[51] The P4 Language Consortium. (2017, May) P4 16 Language Specification. Available at: https://p4.org/p4-spec/docs/P4-16-v1.0.0-spec.html.
[52] Y.-W. Chen, L.-H. Yen, W.-C. Wang, C.-A. Chuang, Y.-S. Liu, and C.-C. Tseng, "P4-enabled bandwidth management," in 2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS). IEEE, 2019.
[53] Campello R.J.G.B., Moulavi D., Sander J. "Density-Based Clustering Based on Hierarchical Density Estimates." In: Pei J., Tseng V.S., Cao L., Motoda H., Xu G. (eds) Advances in Knowledge Discovery and Data Mining. PAKDD 2013. Lecture Notes in Computer Science, vol 7819. Springer, Berlin, Heidelberg, 2013.
[54] S. Makridakis, E. Spiliotis, V. Assimakopoulos, "Statistical and machine learning forecasting methods: Concerns and ways forward," PLOS ONE 13 (3), 2018.
[55] Y. Lv, Y. Duan, W. Kang, Z. Li, F. Wang, Traffic Flow Prediction With Big Data: A Deep Learning Approach, IEEE Transactions on Intelligent Transportation Systems 16 (2), 2015.
[56] F. Kong, J. Li, B. Jiang, H. Song, Short-term traffic flow prediction in smart multimedia system for Internet of Vehicles based on deep belief network, Future Generation Computer Systems 93, 2019.
[57] M. Aqib, R. Mehmood, A. Albeshri, A. Alzahrani, Disaster management in smart cities by forecasting traffic plan using deep learning and gpus, in: International Conference on Smart Cities, Infrastructure, Technologies and Applications, Springer, 2017.
[58] Z. Zhao, W. Chen, X. Wu, P. C. Y. Chen, J. Liu, LSTM network: a deep learning approach for short-term traffic forecast, IET Intelligent Transport Systems 11 (2), 2017.
[59] S. Goudarzi, M. N. Kama, M. H. Anisi, S. A. Soleymani, F. Doctor, Self-Organizing Traffic Flow Prediction with an Optimized Deep Belief Network for Internet of Vehicles, Sensors 18 (10), 2018.
[60] H. Li, Research on prediction of traffic flow based on dynamic fuzzy neural networks, Neural Computing and Applications 27 (7), 2016.
[61] R. Fu, Z. Zhang, L. Li, Using LSTM and GRU neural network methods for traffic flow prediction, in: 2016 31st Youth Aca demic Annual Conference of Chinese Association of Automation (YAC), 2016.
[62] T. N. Sainath, O. Vinyals, A. Senior, H. Sak, Convolutional, long short-term memory, fully connected deep neural networks, in: 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2015.

[63] ETSI Zero-touch Network Service Management group. Available at: https://www.etsi.org/technologies/zero-touch-network-service-management
[64] ENISA, Threat Landscape for 5GNetworks: Updated Threat assessment for the fifth generation of mobile networks (5G), Tech. Rep. December 2020.
[65] J. Ordonez-Lucena, P. Ameigeiras, D. Lopez, J. J. Ramos-Munoz, J. Lorca, J. Folgueira, Network Slicing for 5G with SDN/NFV: Concepts, Architectures, and Challenges, IEEE Communications Magazine 55 (5), 2017.
[66] R. F. Olimid, G. Nencioni, 5G Network Slicing: A Security Overview, IEEE Access 8, 2020.
[67] C. Gong, J. Liu, Q. Zhang, H. Chen, Z. Gong, The Characteristics of Cloud Computing, in: 2010 39th International Conference on Parallel Processing Workshops, IEEE, 2010.
[68] E. M. Hutchins, M. J. Cloppert, R. M. Amin, Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains, Proceedings of the 6th International Conference on Information Warfare and Security, 2011.
[69] MITRE, CVE vulnerability catalog. CVE-2017-0144., Available from MITRE, CVE-ID CVE-2017-0144. (Mar. 2017). Available at: http://cve.mitre.org/cgi-bin/cvename.cgi?name=2017-0144
[70] N. Alomar, P. Wijesekera, E. Qiu, S. Egelman, "You've Got Your Nice List of Bugs, Now What?" Vulnerability Discovery and Management Processes in the Wild, in: Sixteenth Symposium on Usable Privacy and Security (SOUPS 2020), USENIX Association, 2020.
[71] NIRTD, Cybersecurity Game-Change Research & Development Recommendations (2010). Available at: https://www.nitrd.gov/pubs/CSIA_IWG_%20Cybersecurity_%20GameChange_RD_%20Recommendations_20100513.pdf
[72] X. Zhou, Y. Lu, Y. Wang, X. Yan, Overview on Moving Target Network Defense, in: 2018 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC), IEEE, 2018.
[73] A. Aydeger, N. Saputro, K. Akkaya, A moving target defense and network forensics framework for ISP networks using SDN and NFV, Future Generation Computer Systems 94, 2019.
[74] Filebeat lightweight log analysis. Available at: https://www.elastic.co/beats/filebeat
[75] Postman Collaboration Platform for API Development. Available at: https://www.postman.com/
[76] L. Szekeres, M. Payer, TaoWei, D. Song, SoK: EternalWar in Memory, in: 2013 IEEE Symposium on Security and Privacy, IEEE, 2013.
[77] H.-G. Berns, T. Burnett, R. Gran, R.Wilkes, GPS time synchronization in school-network cosmic ray detectors, IEEE Transactions on Nuclear Science 51 (3), 2004.
[78] J. Martin, J. Burbank, W. Kasch, P. D. L. Mills, Network Time Protocol Version 4: Protocol and Algorithms Specification, RFC 5905, Jun. 2010.
[79] IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems, IEEE Std 1588-2019 (Revision of IEEE Std 1588-2008), 2020.
[80] 5Growth "Specification of ICT17 in-house deployment." Deliverable D3.2, April 2020.
[81] J. Baranda, J. Mangues, E. Zeydan, C. Casetti, C. Fabiana Chiasserini, M. Malinverno, C. Puligheddu, M. Groshev, C. Guimarães, K. Tomakh, D. Kucherenko, O. Kolodiaznnyi, Demo: AIML-as-a-Service for SLA management of a Digital Twin Virtual Network Service , in Proceedings of the International Conference on Computer Communications (IEEE INFOCOM), 10-13 May 2021.
[82] ML-Driven Scaling of Digital Twin Service in 5Growth (detailed description - long version). Available at: https://www.youtube.com/watch?v=K5GyrAD7h_Q&t=27s
[83] D. de Vleeschauwer et al., "5Growth Data-Driven AI-based Scaling", in Procs. of the the European Conference on Networks and Communications & 6G Summit (EUCNC'21), 10-12 June 2021.

[84] L. Girletti, M. Groshev, C. Guimarães, C. J. Bernardos and A. de la Oliva, "An Intelligent Edge-based Digital Twin for Robotics," 2020 IEEE Globecom Workshops (GC Wkshps, 2020.
[85] J. Baranda et al., "On the Integration of AI/ML-based scaling operations in the 5Growth platform," 2020 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), 2020.
[86] X. Li et al., "Automating Vertical Services Deployments over the 5GT Platform," in IEEE Communications Magazine, vol. 58, no. 7, pp. 44-50, July 2020, doi: 10.1109/MCOM.001.1900582.
[87] GNS3, The Graphical Network Simulator-3. Available at: https://www.gns3.com/.
[88] P. Trakadas et al., "Comparison of Management and Orchestration Solutions for the 5G Era", MDPI J. Sens. and Actuator Nets., Jan 2020.
[89] Multus container network interface. Available at: https://github.com/intel/multus-cni
[90] Mininet. An instant virtual network on your laptop (or other PC). Available at: http://mininet.org/
[91] J. A. Ayala-Romero, A. Garcia-Saavedra, M. Gramaglia, X. Costa-Perez, A. Banchs and J. J. Alcaraz, "vrAIn: Deep Learning based Orchestration for Computing and Radio Resources in vRANs," in IEEE Transactions on Mobile Computing (Accepted).
[92] ns-3 network simulator, Available at: https://www.nsnam.org/
[93] Multus CNI code repository, Available at: https://github.com/intel/multus-cni
[94] Demonstration of Performance Isolation for 5G Network Slicing, Available at: https://www.youtube.com/watch?v=DSGHUBFuvYs
[95] Cocozza, Massimo, Giovanni Foti, and Fabrizio Arneodo. "SI MO. NE. Innovative System for Metropolitan Area Mobility Management," 16th ITS World Congress and Exhibition on Intelligent Transport Systems and Services, 2009.
[96] V. A. Cunha, D. Corujo, J. P. Barraca, R. L. Aguiar, "TOTP Moving Target Defense for sensitive network services", Pervasive and Mobile Computing (Accepted).
[97] 5Gr-VS OpenAPI specification, Available at: https://github.com/5growth/5gr-vs/blob/master/API/5GR-VS-openapi.yaml
[98] 5TONIC, Available at: https://www.5tonic.org/
[99] Apache Spark, "Apache Spark™ - Unified Analytics Engine for BigData", Available at: https://spark.apache.org/ .
[100] Apache Kafka, "A distributed streaming platform", https://kafka.apache.org/.
[101] Jorge Martín-Pérez, Koteswararao Kondepu, Danny De Vleeschauwer, et al, "Dimensioning of V2X Services in 5G Networks through Forecast-based Scaling", Available at: arXiv:2105.12527.
[102] 5G-TRANSFORMER, "D3.2, Initial Vertical Slicer Reference Implementation," November 2018.
[103] 5Growth, "D5.4: Report on WP5 Progress and Update of CoDEP," May 2021.