H2020 DAEMON Project
Grant Agreement No. 101017109

Deliverable 4.2

Refined design of intelligent orchestration and management mechanisms

Abstract

D4.2 is the second deliverable of WP4 in DAEMON, which builds on D4.1 and presents the detailed results of the identified research threads in the latter. WP4 focuses on functions related to service and resource management and orchestration. It covers Objective 2 (Developing specialized NI-assisted network functionalities for B5G systems) and several KPIs in Objective 4 (Demonstrating the viability and performance of NI-native B5G networks), namely: (K1) VNF energy consumption reduction, (K2) Saving of computational resources at the edge, (K4) OPEX saving, (K5) Reliability, (K7) Anomaly detection recall and sensitivity, (K8) Vertical service response time and (K9) Optimality gap of network management decisions. For that purpose, there are 13 NI solutions which are described in this deliverable, grouped with respect to the four goals that were identified in WP4: (1) Energy-aware VNF orchestration, (2) Capacity forecasting, (3) Automated anomaly response, and (4) Self-learning MANO. It is first discussed how these goals can benefit from each other, how the proposed NI solutions fit into the overall DAEMON architecture (of WP2) and how large experiments are planned to explore then more in WP5. Subsequently, each of these NI solutions is discussed in detail and some initial results are presented, while more through experimentation is planned in WP5. Finally, the conclusions are drawn and future directions and the path toward future deliverables of WP4 are described.

Document properties

Document number	D4.2
Document title	Refined design of intelligent orchestration and management mechanisms
Document responsible	UMA
Document editor	Lidia Fuentes (UMA) Mercedes Amor (UMA)

Editorial team

Partner	Name	Surname	Sections
UMA	Lidia Mercedes Ángel	Fuentes Amor Cañete	1, 2, 3, 4, 5, 6, 2.1, 2.2
IMDEA	Marco Sergi	Fiore Alcalá	3.1, 3.2, 3.3, 4.3, 6.2
WINGS	Sokratis Ioannis Ioannis Evangelos	Barmpounakis Chondroulis Belikaidis Kosmatos	4.1
NEC	Andres Josep Xavier	Garcia Saavedra Salvat	2.3, 2.4, 4.4, 6.1
IMEC	Miguel Paola	Camelo Soto	5.2
TID	Andra	Lutu	4.2, 6.3
TUD	George	Iosidifis	1, 3.4, 6.2, 7
NBL	Danny Chia-Yu Angelos	De Vleeschauwer Chang Pentelas	5.1, 5.2

Target dissemination level	Public
Status of the document	Final
Version	1.0

Production properties

Reviewers	Alexandros Kostopoulos (OTE) Marco Gramaglia (UC3M)

Document history

Revision	Date	Issued by	Description
0.1	05/09/2022	All	Initial Content
0.2	25/09/2022	All	Initial Draft
0.3	10/11/2022	Editors	First version, ready to be reviewed.
0.6	21/11/2022	Editors	Complete version for quality check
1.0	30/11/2022	IMDEA	Final version

Disclaimer

This document has been produced in the context of the DAEMON Project. The research leading to these results has received funding from the European Union Horizon 2020 research and innovation programme under grant agreement no.101017109.

Table of Contents

List of Figures

List of Tables

List of Acronyms

3GPP - 3rd Generation Partnership Project
5G– Fifth Generation
6G – Sixth Generation
AI – Artificial Intelligence
API – Application Programming Interface
AARES - Automated Anomaly Response
B5G – Beyond 5th Generation
BBU - baseband units
BP-vRAN - Bayesian optimization for Power consumption in vRANs
CART - Classification and Regression Tree
CB - Contextual Bandit
CDF - Cumulative Distribution Function (*)
CGP-UCB - Contextual-Gaussian-Process-Bandit-Optimization
CNN – Convolutional Neural Network
CNN-VAE – Convolutional Neural Network Variational Autoencoder (*)
CPU– Central Processing Unit
COTS – Commercial Off The Shelf
CU – Central Unit
DBSCAN - Density-Based Spatial Clustering of Applications with Noise
DDQL - Double Deep Q-Learning
DL – DownLink
DMARL - Deep Multi-Agent Reinforcement Learning
DNN – Deep Neural Network
DQL - Deep Q-Learning
DQN – Deep Q-Network
DU - Distributed Unit
ENI – Essential Node Identifier
ES – Exponential Smoothing
FL– Federated Learning
gNB – next generation Node B
GP - Gaussian Processes
IE - Information Element
IoT – Internet of Things
IQL - Independent Q-learning
iTAREA - efficient TAsk and REsource Allocator
KL – Kullback–Leibler
KNF - Kubernetes-based Virtual Network Function
KPI– Key Performance Indicator
LA - Learning Agent
LSTM – Long Short-Term Memory
MANO – MANagement and Orchestration
MAPE – Monitor, Analyze, Plan and Execute
MAPE-K - Monitor, Analyze, Plan, Execute and Knowledge
MCS – Modulation and Coding Scheme
MDP – Markovian Decision Process
MEC – Multi-access Edge Computing
MILP – Mixed Integer Linear Programming
MSE – Mean Square Error
ML - Machine Learning
MOS – Mean Opinion Score
NFV – Network Function Virtualization
NFVi – Network Function Virtualization Infrastructure

NI - Network Intelligence
NIF – Network Intelligence Function
NN - Neural Network
NS – Network Service
NSSI –Network Slice Subnet Instance
NWDAF – NetWork Data Analytics Function
OM – Operation mode
OPEX – OPerating EXpenses
O-RAN - Open Radio Access Network
OSM – Open-Source MANO
PoP - Point-of-presence
QoS – Quality of Service
QoE – Quality of Experience
RAM – Random Access Memory
RAN – Radio Access Network
ReLU – Rectified Linear Unit
RKHS - Reproductive Kernel Hilbert Space
RNN – Recurrent Neural Network
RIC– RAN Intelligence Controller
RL - Reinforcement Learning
RT – Real Time
RT-RIC – Real Time RAN Intelligent Controller
RTT – Round Trip Time
RU - Radio Unit
SBP-vRAN - Safe Bayesian optimization for Power consumption in vRANs
SCS - Silhouette Coefficient Score
SDN – _Software Defined Networking
SFC - Service Function Chain
SFC-E - SFC embedding
SFC-R - SFC request
SLA – Service Level Agreement
SMO– Service Management and Orchestration
SMT – Satisfiability Modulo Theories
SNR – Signal to Noise Ratio
TES – Thresholded Exponential Smoothing
TES-RNN - Thresholded Exponential Smoothing-Recurrent Neural Network (*)
UE – User Equipment
UL – UpLink
VAE – Variational Autoencoder
vBS - virtualized Base Station
VIM – Virtual Infrastructure Manager
VM – Virtual Machine
vRAN – virtualized Radio Access Network
VNF – Virtual Network Function
VNF-PC – Virtual Network Function placement

Executive summary

This deliverable reports the progress made by DAEMON on the design and development of NI-assisted functionalities for Beyond fifth Generation (B5G) mobile networks, with a focus on the area of service and resource management and orchestration. To achieve this overarching goal, we study four key B5G functionalities: (i) energy-aware Virtual Network Function (VNF) orchestration, (ii) capacity forecasting, (iii) automated anomaly detection, and (iv) self-learning management and orchestration.

D4.2 builds on the results of D4.1 and presents the latest developments of DAEMON on the identified research threads for each of the above functionalities. In detail, Section 1 introduces the main results of this deliverable and explains their connection to the previous deliverable (D4.1) as well as to the other work packages. The rest of the sections focus on each of the building blocks, and for each solution it is reported which specific requirements from D2.2 are satisfied, how it relates to D2.2 guidelines, which standards are related to it, and how it can be encoded using the MAPE-K representation. This systematic presentation approach allows the reader to understand and compare the proposed solutions in a unified way.

Namely, Section 2 presents the results related to the proposed solutions for Energy-aware VNF orchestration. These solutions are designed to work at different levels of a virtualized next-generation mobile network and specifically are related to (i) placement and autoscaling of virtualized NFs at user/data level (Section 2.1and Section 2.2) and (ii) virtualized NFs at the RAN (Section 2.3 and Section 2.4). The solutions leverage a diverse set of algorithms and AI mechanisms, ranging from Deep Learning to prediction-assisted VNF placement and scaling and to Bayesian learning techniques for the design of meta policies in O-RAN systems. The targeted optimization metrics go beyond energy savings all the way to throughput maximization and service-specific KPIs such as inference accuracy in edge analytic services. Therefore, the proposed solutions not only reduce the energy costs, but are able to trade off energy consumption with performance. The results of this section have been published in scientific conferences and journals and have been verified through full-scale implementation in testbeds and/or data-driven evaluations that are reported in the deliverables of WP5.

Section 3 continuous with the solutions related to capacity forecasting methods. It presents the results of anticipatory networking methods where resources are reserved or allocated before the actual demand (or traffic) is realized. In this context, D4.2 presents four solutions: (i) anticipatory allocation of resources, (ii) Virtual Machine (VM) reservation in a network core datacenter, (iii) minimization of video streaming slice OPEX, (iv) and prediction-assisted network resource management. The solutions that are employed to achieve these tasks include time-series forecasting models that merge classical exponential smoothing techniques with Recurrent Neural Networks, meta-learning techniques that identify the most suitable loss function using the problem's data, and hybrid learning solutions that combine offline-trained predictors and online optimization techniques towards fast and robust resource allocation decisions. The results of this section address several functionality requirements from D2.2 and go well-beyond the state-of-the-art as they reveal and address several limitations of classical AI/ML techniques, such as the slow convergence of online learning algorithms or the suboptimal performance of predefined loss functions.

Section 4 presents the progress of DAEMON on the topic of anomaly detection. We present four solutions that tackle anomalies in different parts of the cellular ecosystem, from the RAN and base stations to the interconnection between mobile operators across different countries. In detail, the first solution employs federated learning in order to perform clustering using, without sharing, local data across different agents that are deployed into different network locations. The second proposal integrates advanced deep learning models for anomaly detection in IoT devices for different verticals, using only control plane information. Following that, D4.2 reports results on anticipatory anomaly detection, where an alarm is triggered whenever an abnormal traffic load is expected for a specific Network Slice. The fourth and final solution explores the problem of noisy neighbor in computing environments implementing VNFs of virtualized base stations, and presents experimental evidence of this issue and of its effects on network performance. The assessment of these methods is conducted using a range of datasets and experiments, hence allowing their realistic evaluation. These results are reported in the respective work packages.

Finally, Section 5 presents self-learning MANagement and Orchestration (MANO) solutions which are crucial in B5G networks given the range and complexity of the decisions involved in their operation. The first group of solutions studies centralized and distributed Reinforcement Learning (RL) techniques for VNF placements across different network locations toward supporting a newly-deployed network service. This builds upon and extends the respective results reported in D4.1 The second group of solutions considers the problem of automated scaling of allocated network resources in order to meet the time-varying demands of users. To that end, we use two different techniques, namely a control-theoretic solution and an RL-based solution which track the number of consumed resources and increase or decrease the VNF instances accordingly. The evaluation and comparison results are presented in D5.2.

The deliverable concludes in Section 6 by discussing the proposed solutions through the lens of WP2. This way, we provide feedback to WP2 on the applicability of the architecture and in particular we identify two cases where the proposed solutions need to be aligned in order to maximizer the accrued benefits. We identify two such cases, namely the energy-aware VNF placement and vRAN orchestration, and the capacity forecasting and anomaly detection, and we use a detailed MAPE-K representation in order to align them and exploit their complementarity.

1 Introduction

The main goal of this deliverable is to report WP4 advances towards the achievement of its main objective, which is to design NI-based solutions for the functionality related to service and resource orchestration and management. WP4 particularly covers DAEMON's objective 2 (Developing specialized NI-assisted network functionalities for B5G systems), and several specific Key Performance Indicators (KPIs) of objective 4 (Demonstrating the viability and performance of NI-native B5G networks). In this deliverable, we provide an overview of the solutions proposed by the DAEMON consortium from the beginning of the project and the interrelationships between them, as well as we present some mappings with the requirements and architecture proposed in D2.2 [1].

This deliverable is organized as follows. First, in this section (subsection 1.1), we show and describe the overall picture with the main solutions, locating them in different domains along the path between the mobile network users and the rest of the Internet. In subsection 1.2, we present tables that summarize the fulfillment of the requirements reported in D2.1 [2], as they are updated by D2.2 [1], and explain how these are addressed by each solution.

The rest of the sections are devoted to describing each contribution in depth. Each of these subsections is structured as follows:

- Requirements in D2.2: presents the mapping between the requirements defined in D2.2 [1] to the functionalities of each D4.2 solution reported here.
- Guidelines from D2.2: explains implementation and design decisions made by each solution, in order to accomplish the guidelines established in D2.2 [1], Section 4.2.7.
- Solution description: describes the approach followed by a concrete solution and also the specifics of its internal implementation and detailed design. For those solutions that were totally described in D4.1 [3], only a brief summary is included. The achieved advances and new results are reported in detail in this section. On the other hand, one problem statement is reported for the first time in Section 4.1.3.
- N-MAPE-K representation: each subsection of this type describes the mapping between the solution and the N-MAPE-K architecture presented in D2.2 [1].

Therefore, we consider that the traceability between requirements, architecture (WP2) and detailed design of each solution developed as part of WP4, can be easily followed thanks to the mapping definition subsections. We include all the solutions developed by DAEMON partners from the beginning of the project, including one problem description. So, even those solutions that have finished at the end of year one were incorporated into this document completing their description with the mapping to requirements, guidelines, and the N-MAPE-K representation.

After all the solutions are described, Section 6 focuses on reconciling the WP2 and WP4 perspectives. We have tried to accommodate the architectural viewpoint represented by the N-MAPE-K with the detailed design of each solution. We have identified some scenarios that illustrate how some WP4 solutions can be coordinated to achieve a common goal (i.e., intelligent resources or service orchestration) or that some conflicts can also arise when two or more solutions work together. Finally, we discuss the deliverable contributions in a conclusion section.

1.1 Overview of D4.2 solutions

Figure 1 shows the big picture of D4.2 main contributions, which are organized mainly in two layers: ***resource orchestration*** and ***service orchestration***. We will explain this figure from the bottom to the top. The bottom of the figure is the same that appears in Annex I of the DAEMON proposal, showing the concept of the DAEMON NI-native architecture. We use this part of the figure to show the different domains at which the algorithms of each network intelligence solution impact more, considering the different parts of the network from the subscriber to the rest of the Internet. Since DAEMON tackles the problem of resource management in flexible virtualized networks, most of the proposals presented here perform the orchestration of resources and services above a virtualized infrastructure, hiding the hardware details of heterogeneous networks. This is then represented in the figure as a sublayer "Virtualized infrastructure" inside the network infrastructure.

Each of the solutions proposed at the resource and service orchestration layers is represented as a small house, named with the section number (e.g., S2.1 stands for section 2.1) and a phrase summarizing its main functionality. For those solutions that try to optimize some KPIs, each house includes a roof with icons that represent the optimization goals considered by that work (i.e., energy, performance and/or OPEX). In the right hand of the figure, a simplified version of the N-MAPE-K architecture proposed in D2.2 [1] is represented to show that HW/virtual resources are orchestrated by different artificial intelligence algorithms, and the same happens with the service orchestration.

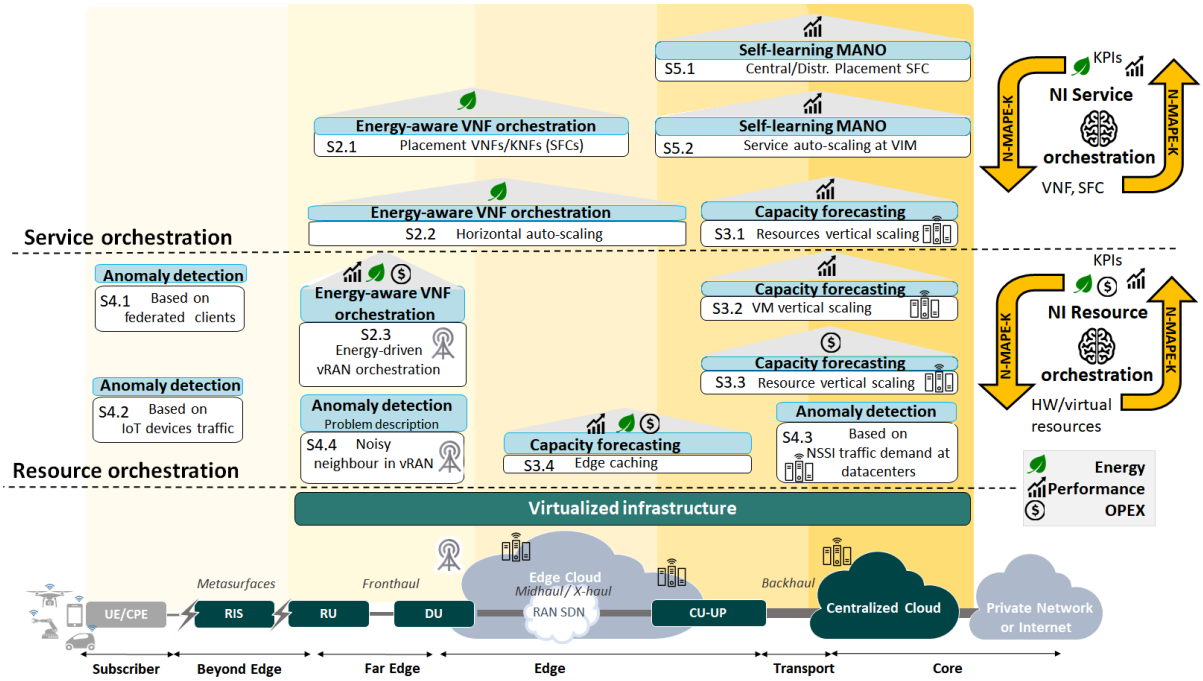

Figure 1. *Summary and overall positioning of D4.2 main contributions.*

Ten solutions address intelligent ***resource orchestration*** from different points of view, some of them complementary and others alternatives. The problem of energy-aware VNF orchestration (WP4, task 4.1) of network resources is tackled by two approaches described in S2.2 and S2.3. The solution S2.2 performs energy-aware horizontal auto-scaling of network resources encompassing many Points-of Presence (PoPs) from subscribers' devices to Edge and Cloud equipment, guaranteeing there are enough virtual machines to accommodate the VNFs needed by the current network workload. The solution S2.3 focuses on saving energy inside a virtualized base station.

The partners of DAEMON propose four solutions on capacity forecasting approaches (WP4, task 4.2) that anticipate the optimal amount of resources that need to be allocated inside network entities at each reconfiguration opportunity. Two of these solutions perform vertical scaling of HW resources like computation and network BW inside a datacenter or network equipment (S3.1 and S3.3), another work anticipates the virtual machine demands inside for example a datacenter (S3.2), and finally, one solution improves the resource orchestration by caching functionalities at edge servers or MEC systems (S3.4).

Regarding the anomaly detection works (WP4, task 4.3), one of them is focused on detecting anomalies based on NSSI traffic demands at datacenters, and another work (S4.3) analyses the IoT devices traffic to detect some abnormal situations that need to be reported. The work presented in S4.1 proposes a solution based on federated clients that monitors own compute resources (CPU, memory) and network-related KPIs (DL/UL throughput, latency, packet error rate, jitter) to infer general anomaly detection rules. Lastly, S4.4 focuses on resource orchestration inside a virtualized base station, but from the point of view of anomaly detection considering the noisy neighbor's effect. This is crucial if we want to optimize a virtualized Radio Access Network (vRAN) system in noisy environments. This problem is described later, and solutions will be proposed during the next year.

At the ***service orchestration*** layer, one solution (S2.1) proposes an energy-aware VNF orchestration algorithm to perform the optimal placement of VNFs implemented as KNFs (Kubernetes Network Functions), encompassing user, network and edge devices, and also datacenters. The self-learning MANO work S5.1 proposes two versions of a service placement algorithm (i.e., central and distributed) of Service Function Chains (SFCs) considering a number of geographically distributed datacenters (also referred to as Points-of Presence (PoPs)). On the other hand, the S5.2 solution performs service auto-scaling, calculating the number of VNF replicas in an NFV environment.

1.2 Summary of requirements satisfaction by D4.2 solutions

In this subsection, we analyze how the NI solutions proposed in WP4 satisfy the functional and non-functional requirements defined in D2.1 [2] and updated in D2.2 [1], showing clearly the relationships between the activities of WP2 and WP4. The following tables summarize the satisfaction of requirements related to four clusters of network functionalities defined in D2.1: energy-aware VNF placement (Table 1), capacity forecasting (Table 2), automated anomaly response (Table 3), and self-learning MANO (Table 4).

The methodology followed for gathering and documenting requirement satisfaction is as follows. Authors of each solution made a list of the requirements addressed by the concrete solution and provided a rationale for that (see subsections Requirements in D2.2 [1]). Editors analyzed these lists and proposed a mapping, which was revised by deliverable reviewers, task and WP4 leaders. After several iterations, the results are shown here.

For each group of requirements (e.g., FR-EAWVNF), we put solutions that satisfy all the requirements of such group in columns (e.g., column S2.1). The last column (labeled D3.2) indicates if there is a solution presented in deliverable D3.2 that also satisfies a requirement. Each solution is identified by the section where is described in this document. The last column (labeled D3.2) indicates if the requirement is satisfied by any solution presented in deliverable 3.2. Each row refers to a functional (FR) or non-functional (NFR) requirement included in D2.2 [1], using the same identifier. If a solution is working on fulfilling a given requirement but it does not meet it yet, the check is grey. For each requirement satisfied, a description of how the NI functionality solution meets it is included in the corresponding subsection.

From the data collected, we can derive that many of the (functional and non-functional) requirements related to WP4 tasks are already covered. These clusters enclose 35 requirements, most of which are satisfied. Those whose satisfaction is not currently addressed by any of the solutions presented in this deliverable 4.2 are intended to be satisfied as part of the work of year 3 of the project.

Simply looking at each table is possible to have an overview of which solution(s) satisfy which requirement and if there is some pending requirement. This mapping, expressed by a set of tables, is also of great value in evaluating WP2 activities and decisions.

***Table 1**. Energy-aware VNF placement requirements satisfaction of D4.2 contributions.*

			S2.1	S2.2	S2.3	D3.2
FR-EAWVNF-000 DAEMON Energy-aware VNF placement (EAWVNF) shall profile the energy footprint of those network tasks that influence the network's global power consumption.			✓	✓	✓	✓
	FR-EAWVNF-001		✓			
		FR-EAWVNF-001.00	✓	✓		
		FR-EAWVNF-001.01	✓			
		NFR-EAWVNF-001		✓		
		NFR-EAWVNF-002	✓	✓		
		NFR-EAWVNF-003	✓			
	FR-EAWVNF-002		✓			
	FR-EAWVNF-003		✓			
		FR-EAWVNF-003.00		✓		
		FR-EAWVNF-003.01		✓		
	FR-EAWVNF-004		✓	✓		
		FR-EAWVNF-004.00	✓	✓		
		FR-EAWVNF-004.01	✓			
	FR-EAWVNF-005			✓	✓	✓
		NFR-EAWVNF-004			✓	✓
		NFR-EAWVNF-005			✓	✓
		NFR-EAWVNF-006		✓	✓	
	FR-EAWVNF-006			✓		
		FR-EAWVNF-006.00				
		FR-EAWVNF-006.01		✓		

***Table 2**. Capacity forecasting requirements satisfaction of D4.2 contributions.*

			S3.1	S3.2	S3.3	S3.4	S4.3	D3.2
FR-CFORE-000 DAEMON shall design capacity forecast models that can support NI algorithms across the mobile network architecture			✓	✓	✓	✓		
	FR-CFORE-001							
	FR-CFORE-002		✓	✓	✓			
	FR-CFORE-003					✓		
	FR-CFORE-004						✓	✓
	FR-CFORE-005			✓	✓	✓		

		FR-CFORE-006						
		FR-CFORE-007						

***Table 3**. Automated anomaly response requirements satisfaction of D4.2 contributions.*

			S4.1	S4.2	S4.3	D3.2
FR-AARES-000 DAEMON shall automatically detect, analyze, and act against anomalous behaviors.			✓	✓	✓	✓
	FR-AARES-001		✓	✓		
	FR-AARES-002			✓	✓	✓
	FR-AARES-003		✓	✓		
	FR-AARES-004			✓		
		NFR-AARES-000				

***Table 4.** Self-learning MANO requirements satisfaction of D4.2 contributions.*

			S5.1	S5.2	D3.2
FR-SLMANO-000 DAEMON controllers and orchestrators should be steered by high-level QoE targets and business KPIs (high-level intents) rather than strict QoS goals and technical KPIs.			✓	✓	✓
	FR-SLMANO-000				
	FR-SLMANO-002		✓		
		FR-SLMANO-002.00			
	FR-SLMANO-003		✓	✓	✓
		NFR-SLMANO-001			
	FR-SLMANO-004		✓		
	FR-SLMANO-005				
	FR-SLMANO-006				
	FR-SLMANO-007				

2 Energy-aware VNF orchestration

Energy-aware VNF orchestration aims at managing VNF-demanding resources and services. The solutions presented in this task focus on the energy-aware orchestration of services at different levels of a virtualized next-generation mobile network. In the following, we present three solutions that deal with virtualized services orchestration at two different levels: placement and autoscaling of virtualized NFs at user/data level (Section 2.1 and Section 2.2); and virtualized NFs at the RAN (Section 2.3 and Section 2.4).

2.1 Energy-aware VNF placement in Open-Source MANO infrastructures

The solution supports VNF placement in heterogeneous edge infrastructures by considering the energy consumption as well as the computation and communication delays within a Network Function Virtualization Infrastructure (NFVI). The proposed solution has the specificity of considering requirements that are not commonly considered in VNF placement decisions, such as the available Random-Access Memory (RAM), storage, or specific hardware/server configuration. Based on the different constraints above, the solution allows selecting the nodes where VNFs can be run so as to both reduce energy consumption and meet the needs of the infrastructure. The solution has been integrated as an extension of the Open-Source Management and Orchestration (OSM) [4] project.

2.1.1 Requirements in D2.2

The mapping between the requirements established for energy-aware VNF placement in D2.2 [1] and the proposed model is presented in the following *Table 5*.

***Table 5** . Energy-aware VNF placement. Requirement satisfaction.*

Requirement from D2.2 [1]	Rationale
FR-EAWVNF-000	This NI Functionality uses specific models of the power consumption of VNFs to optimize their placement in a heterogeneous infrastructure.
FR-EAWVNF-001	This NI Functionality monitors the power and resource consumption profile of VNFs to optimize their placement in a heterogeneous infrastructure.
FR-EAWVNF-001.00	This solution explicitly includes the number of CPU cycles in the energy consumption model of a VNF (See equation in section 2.1.3.1).
FR-EAWVNF-001.01	This solution explicitly includes the cost of data communication in the energy consumption model of a VNF (See equation in section 2.1.3.1).
FR-EAWVNF-002	The solution monitors the impact of hardware resource usage by VNFs in the estimation of the energy footprint. The solution also considers the computation, communication, and storage resources as part of the placement algorithm, which uses a Satisfiability modulo theories solver.
FR-EAWVNF-004.00	The proposed solution has the specificity of an energy profile considering requirements that are not commonly considered in VNF placement decisions, such as the available Random Access Memory (RAM), storage, or specific hardware/server configurations. The proposed solution has the specificity of considering in the energy profile the dependency relationships of VNF location requirements that are not commonly considered in VNF placement decisions by means of a variability model of the network infrastructure and a variability model of software infrastructure. Based on the different constraints above, the solution allows selecting the nodes where VNFs can be run so as to both reduce energy consumption and meet the needs of the infrastructure.
FR-EAWVNF-004.01	The solution considers the energy cost of computing VNF placement, which is considered part of the global energy footprint of the solution. We address this requirement as part of the functions of the Energy Efficient Task Resource Allocator, which considers the computational and communication energy consumption of VNFs based on their location in the infrastructure to place them in an energy-efficient manner.
NFR-EAWVNF-002	The placement solution reduces up to 51% of the energy consumption compared with the default deployment proposed by the existing MANO standard solution.

2.1.2 Guidelines from D2.2

The design of our solution for energy-aware VNF placement follows the guidelines established in D2.2 [1], Section 4.1, to tailor energy and latency models to the specificities of network environments; in particular, the approach takes into account the fact that the computational complexity of an NI model has a strong correlation with its latency performance and energy consumption.

Having this guideline in mind, our placement algorithm, named iTAREA and presented in detail in the next section, contains a variable that limits the search range of the algorithm to solutions that are feasible in practical network settings. In addition, the performance of the placement solution has been taken into consideration by evaluating the scalability and, therefore, applicability and complexity of our solution via measurements of the execution time of the placement algorithm for different problem sizes. Finally, our design has been implemented in two different technologies (solvers) in search of the most efficient and scalable solution.

2.1.3 Solution description

In D4.1 [3], Section 3.1, it was described the problem of energy-aware VNF placements (as an initial step to VNF orchestration) by (i) modeling VNFs as a set of reusable service functions, including the definition of resource demands, energy cost and dependencies on the hardware and software of NFV and physical infrastructures; and, (ii) by automating the optimization of VNF orchestration in a heterogeneous mobile infrastructure to reach a certain quality of service (QoS, e.g., minimizing power consumption). The solution makes use of OSM and Kubernetes functionalities for managing VNFs and KNFs deployments, automating and optimizing the deployment process to minimize energy consumption.

Our energy-aware VNF placement solution manages heterogeneous edge infrastructures and considers the energy consumption and the computation and communication delay within an NFVI. The proposed solution also considers requirements that are not commonly considered in the VNF placement problem (VNF-PC), such as RAM, storage, or a specific hardware/server configuration (e.g., a GPU, a specific network card, etc.), and allows energy-efficient placement of VNFs in the infrastructure considering the functional and non-functional requirements of the VNFs. The solution has been integrated as an extension of the OSM project [4] and tested in real-world scenarios.

The overall VNF placement solution, named HADES [5], extends OSM by enabling the placement and configuration of VNFs/KNFs[1] and their subsequent resource allocation and deployment at the edge, minimizing energy consumption and ensuring the QoS. Additionally, HADES allows the suppression and modification of deployments at runtime, NFVI status monitoring, and the deployments considering several Virtual Infrastructure Managers (VIMs) (which are not natively supported by OSM). Furthermore, the use of Kubernetes endows the infrastructure with fault tolerance and high availability.

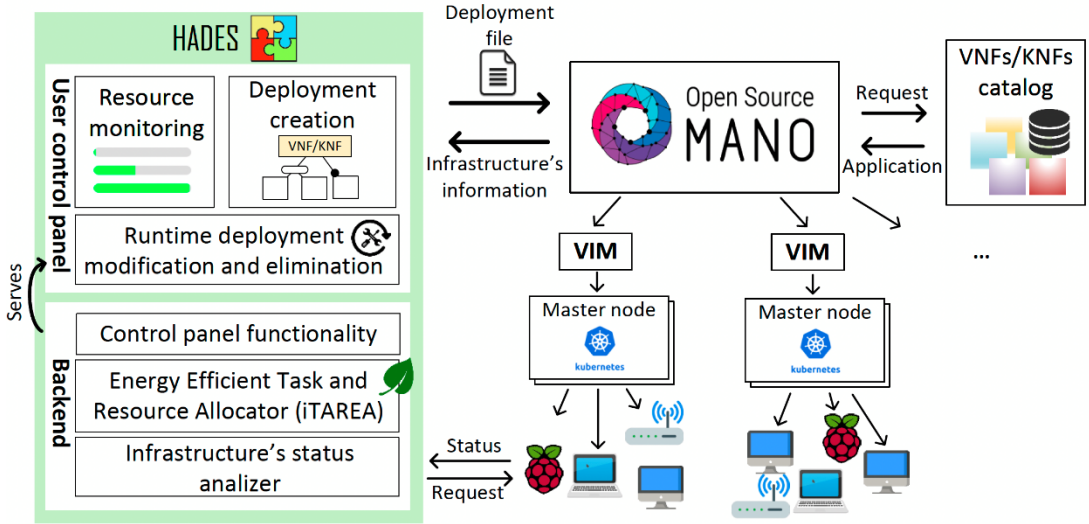

Figure 2. *General overview of HADES.*

Figure 2 shows a general overview of the solution, also showing its integration with OSM. HADES is a web-based solution composed of two main parts: i) the backend, a web-based module that implements the main functionalities, and ii) the control panel, providing the infrastructure administrator with a graphical

[1] OSM supports container technology (via Kubernetes [ref]) to enable the design and implementation of Kubernetes-based Network Functions (KNFs) running inside containers.

interface. The latter interface allows: choosing the VNF/KNF to be deployed and configuring its characteristics; modifying and removing existing deployments; monitoring the status of the infrastructure resources in real-time. The solution collects and uses information on the infrastructure (e.g., node addresses) provided by the OSM and sends OSM-understandable deployment files to the OSM to carry out the deployment.

2.1.3.1 Energy consumption and latency models

The energy-efficient task and resource allocator (iTAREA) module, located in the backend in *Figure 2*, estimates the energy consumption and latency of the application execution towards the goal of minimizing energy consumption and ensuring high QoS standards. The following equations show the expressions used to calculate the computation time ($tCompu_{t,i}$, in seconds) and data transmission time ($tCommu_{i,j,n,z,ct}$, also in seconds) respectively:

$$tCompu_{i,n} = \frac{w_i}{CPU_n v_{i,n}}$$

$$tCommu_{i,j,n,z,ct} = \left(\frac{c_{i,j} + c_{i,j} pR}{Min\left(R_{n,ct}^{Tx}, R_{z,ct}^{Rx}\right)} + t_{n,z}^{prop} \right)$$

where the computation time of KNF i in node n is obtained as the ratio between the number of CPU cycles (CPU_n, collected using tools like iPerf[2]) required by the KNF i and the CPU frequency of the node per core (F_n) multiplied by the percentage of CPU allocated to the task on the node $v_{i,n}$. Meanwhile, the communication time $tCommu_{i,j,n,z,ct}$ is calculated as the ratio of the amount of data to be transmitted between KNF i and KNF j ($c_{i,j}$), being pR the probability of retransmission due to packet loss, and the transmission/reception speed (using connection type ct, e.g., WiFi 2.4, 5G). The propagation delay $t_{n,z}^{prop}$ is calculated as half of the mean round trip time (RTT) obtained by pinging from *n* to z.

Energy consumption is estimated using the expression given in Section 3.1.3.1, in D4.1 [3]. While the energy consumption in the nodes is influenced by several factors, e.g., allocation of storage and RAM, the CPU usage is the most influential factor [6], the expression thus includes the CPU frequency at which the node works: specifically, in cases where solutions that limit the node resources (e.g., VMs, containers) are used, the frequency is the one associated with that VM or container. This approach provides an accurate energy consumption for nodes with software-limited frequency.

The expression in Section 3.1.3.1 of D4.1 [3] also includes a weight that allows capturing the importance of energy savings in each node. This makes it possible to adapt the deployment solution to the needs of the infrastructure, unlike other approaches that have a set of nodes on which the reduction of energy consumption is focused [7]. This weight-based system also provides flexibility to the deployment solutions and eases the definition of policies according to the infrastructure necessities: a practical example is prioritizing the execution of VNFs on devices powered by solar panels, therefore reducing the whole carbon footprint of the infrastructure.

2.1.3.2 Algorithm for energy-aware VNFs placement

As aforementioned, the iTAREA module decides where will be executed each task, as well as the amount of resources (CPU, RAM, storage and bandwidth) allocated to each such task. As the CPU depends on the computing capacity of the node to which a task is assigned, the variable $CPUPartitions$ contains the portions of CPU that can be assigned to each KNF by each core; this reduces the search range (and therefore the problem complexity). Therefore, if this variable has a value of 10, the CPU will be allocated in 10% portions (0.1 cores). Thus, the minimum amount greater than 0 of CPU to allocate would be 0.1 cores, followed by 0.2 cores, and so on.

This module receives as input the information of the nodes (set N), the tasks (τ), the nodes to exclude ($nodesToAvoid$), the set of time restrictions (Tr), and the CPU portions ($CPUPartitions$). Concerning QoS assurance, applications have sets of KNFs with time constraints (i.e., they must be completed within a maximum time), which are part of the problem constraints. As output, it returns the task assignment and the number of resources that the node must reserve for each assigned KNF.

The algorithm of the iTAREA module, showed Figure 3, does the following: first, it excludes the nodes to be avoided from the set N(1); (2) checks that each task has a node assigned; (3-5) ensures that nodes have RAM, disk and bandwidth enough to execute their assigned tasks, while (6-8) assuring to assign enough resources to properly execute each KNF; (9-10) checks that nodes do not allocate more CPU than available (limiting the search range according to the $CPUPartitions$ variable); (11) checks that the nodes fulfill the KNF requirements (in terms of peripherals and sensing units); (9) ensures that the nodes that have to send/receive data between them are using the same communication capacity; (12) verifies

[2] iperf tool. https://iperf.fr

the fulfillment of the time restrictions, considering execution and communication times; finally, the iTAREA returns the solution (KNF assignment, RAM, storage, bandwidth and CPU assigned to each KNF).

As the iTAREA module has been implemented using Z3, a satisfiability modulo theories (SMT) [8] solver, the algorithm always returns a solution (differently from heuristic algorithms), which guarantees that the deployment is feasible or the impossibility of deploying the application if no solution is found. In addition, a large number of constraints (necessary to solve the problems at hand) helps SMT solvers to reduce the search space and find the optimal solution more quickly. In addition, we have also developed another version of the module using the *Gurobi Parallel Mixed Integer Programming* solver. While both SMT solvers and MILP allow handling the variables required by the problem at hand, always return a solution and are optimal, MILP solvers return the solution faster than SMT [9]. Furthermore, Gurobi supports multi-threading, which aids problem scalability.

Input: $N, Tr, \tau, nodesToAvoid, CPUPartitions$

Minimize: $energy \rightarrow \forall(n,m)\epsilon N, i\epsilon\tau : x_{i,n}eCompu(i,n) + \sum_{j\epsilon\tau}(h_{i,j}eSend(c_{i,j},n) + x_{j,m}eRecp(c_{i,j},m))$

Subject to:

$$N = N \setminus nodesToAvoid \quad (1)$$

$$\forall i\epsilon\tau : \sum_{n\epsilon N} x_{i,n} = 1 \quad (2)$$

$$\forall n\epsilon N : \sum_{i\epsilon\tau} x_{i,n} i_{RAM} \leq n_{RAM} \quad (3)$$

$$\forall n\epsilon N : \sum_{i\epsilon\tau} x_{i,n} i_{disk} \leq n_{disk} \quad (4)$$

$$\forall n\epsilon N : \sum_{i\epsilon\tau} x_{i,n} i_{bandwidth} \leq n_{bandwidth} \quad (5)$$

$$\forall n\epsilon N, i\epsilon\tau : x_{i,n} RAM_{i,n} \geq i_{RAM} x_{i,n} \quad (6)$$

$$\forall n\epsilon N, i\epsilon\tau : x_{i,n} disk_{i,n} \geq i_{disk} x_{i,n} \quad (7)$$

$$\forall n\epsilon N, i\epsilon\tau : x_{i,n} bandwidth_n \geq i_{bandwidth} \quad (8)$$

$$\forall n\epsilon N : \sum_{i\epsilon\tau} x_{i,n} CPU_{i,n} \leq CPUPartitionsCores_n \quad (9)$$

$$\forall n\epsilon N, i\epsilon\tau : CPU_{i,n}\epsilon\mathbb{N} \quad (10)$$

$$\forall n\epsilon N, i\epsilon\tau : x_{i,n} \wedge meets(n,i) \vee \neg x_{i,n} \quad (11)$$

$$\forall tr\epsilon Tr, (n,m)\epsilon N, i\epsilon\tau : x_{i,n} tCompu(i,n) + \sum_{(j)\epsilon tr} h_{i,j} x_{j,m} tCommu(c_{i,j},n,m) \leq time_{tr} \quad (12)$$

Return: $\forall i\epsilon\tau, n\epsilon N : x_{i,n} \rightarrow RAM_{i,n}; x_{i,n} \rightarrow disk_{i,n}; x_{i,n} \rightarrow bandwidth_{i,n}; x_{i,n} \rightarrow CPU_{i,n}$

***Figure 3**. Algorithm of the iTAREA module.*

2.1.3.3 Resource monitoring

A specific module of the backend of HADES (labeled as "infrastructure's status analyzer" in *Figure 2*) is responsible for tracking the status of the infrastructure by analyzing monitored data. Its functions are to check the number of VIMs, their characteristics, and their associated cluster (if any); and to obtain real-time information about the workload of the CPU, RAM, disk, and bandwidth of the nodes.

This module has two main functions. On the one hand, it allows the iTAREA module to perform its function of efficient allocation of tasks and resources. In fact, this module is a key part of the proper functioning of the iTAREA module. On the other hand, it allows HADES to be aware of the most demanded nodes or of those that should be excluded from some deployments. This concept is based on the premise that some VNFs/KNFs will need features presented in some nodes (e.g., a specific network card), while for other applications, such a node is used only for computation. In this way, the resources of these nodes remain available for the tasks that can only be executed on them. This process can be carried out automatically by HADES if it detects that a particular VNF needs to be executed on a particular node(s) or included as a rule by the infrastructure administrator.

Regarding the module implementation, it is a process that periodically makes requests to the Kubernetes clusters to know their status in real-time. This information is stored at the backend, which keeps a log. The maintenance of status information over time allows the processing of this data, which makes it possible to apply auto-scaling techniques. An experimental evaluation of HADES will be reported in D5.2.

2.1.4 Mapping to major standards (if applicable) and gaps identification

HADES extends OSM bringing the following capabilities:

- Minimization of energy consumption: HADES minimizes the energy consumption of the deployments.
- Generation of application configuration: HADES allows the user to select both application and QoS-related features prior to deployment.
- QoS assurance: the selected functional and non-functional characteristics of the configured application are considered in the deployment phase.
- Resource allocation optimization: HADES establishes a minimum and maximum range of resource usage (e.g., CPU) that ensures QoS and minimizes resource waste.
- Deployments considering the entire infrastructure: in infrastructures with more than one VIM, HADES allows deployments considering all of them. This functionality is not natively available in OSM.
- Easy deployments modification: HADES eases the deployments modification, keeping intact those non-modified application parts and maintaining consistency with the rest of the VNFs.

2.1.5 N-MAPE-K representation

The following table contains the decomposition of the energy-aware placement solution for OSM into the different components of the N-MAPE-K.

Table 6. *N-MAPE-K loop mapping of energy-aware placement solution.*

N-MAPE-K module	Solution module
Monitor	A resource monitoring module monitors the status of the infrastructure resources (CPU, memory, bandwidth) in real-time.
Analyze	The current status of the infrastructure is analyzed to check the nodes that can meet each KNF/VNF requirement and demanded resources (CPU, RAM, disk and bandwidth) that need to be allocated. The energy footprint of the assignments is evaluated to provide deployments that minimize energy consumption.
Plan	The iTAREA module decides the optimal (energy-aware) placement of each KNF/VNF and the number of resources (CPU, RAM, disk and bandwidth) allocated to this end. The optimal placement and resource allocation is generated.
Execution	The OSM's North Bound Interface (NBI) API is used to communicate the automated placement and deployment of KNF/VNFs with OSM. OSM NBI is RESTful and follows the ETSI SOL005 standard.
Knowledge	KNF/VNF repository, Energy Model, Infrastructure's status, software and hardware infrastructure characteristics (CPU power, peripherals, memory, etc.).
Training Loss / State – Actions – Rewards	The iTAREA module is responsible for determining the assignment of tasks to nodes and the amount of resources to allocate to meet certain QoS, considering the requirements of the applications and the characteristics of the infrastructure. To do so, applications and infrastructure are analyzed and the equations in Section 2.1.3.1 are applied to estimate the energy consumption. As the number of deployments increases, the QoS obtained is compared with the expected QoS, and increasingly precise solutions can be obtained.

2.2 A proactive auto-scaling solution for energy-aware VNF placement in edge-based infrastructures

Auto-scaling allows to manage and improve infrastructure resource availability while optimizing energy consumption when the services' demand or workload varies dynamically. Horizontal auto-scaling, which enables increasing/decreasing the number of nodes of edge infrastructure, allows reducing the energy consumption of the nodes themselves (idle energy consumption).

Our proposed solution is a proactive horizontal auto-scaling framework tailored to edge infrastructures with different operation modes. This solution considers both the base (idle) and dynamic (due to application execution) energy consumption of the nodes, as well as the energy consumption of node scaling. Specifically, the proposed auto-scaling solution has four resource reservation policies, applicable to two different operation modes (i.e., a total of eight configurations), each of which can be used with the desired VNF assignment policy.

2.2.1 Requirements in D2.2

The mapping between the requirements established for the proactive auto-scaling solution for energy-aware VNF placement in edge-based infrastructures in D2.2 [1] is presented in the following table.

***Table 7**. Proactive autoscaling solution and VNF placement. Requirement satisfaction.*

Requirement from D2.2 [1]	Rationale
FR-EAWVNF-000	This NI Functionality uses the power consumption profile of VM and VNFs to optimize infrastructure scaling for VNF placement.
FR-EAWVNF-001.00	The energy consumption model (Section 2.2.3.1) calculates the energy footprint of VNFs in terms of CPU usage according to the node in which VNFs are deployed.
FR-EAWVNF-001.00	The energy consumption model calculates the energy footprint of VNFs in terms of CPU data transmission according to the node in which VNFs are deployed.
FR-EAWVNF-003.00	Our energy consumption model can be used to calculate the energy footprint of VNF migration due to virtualization costs.
FR-EAWVNF-003.01	Our energy consumption model can be used to calculate the energy footprint of VNF migration due to transmission costs.
FR-EAWVNF-006.00	Our energy consumption model can be used to calculate the energy footprint of VNFs vertical scaling.
FR-EAWVNF-006.01	This NI Functionality considers the energy consumption of horizontal scaling to optimize VNF placement. This solution considers both the base (idle) and dynamic (due to application execution) energy consumption of the nodes, as well as the energy consumption of node scaling.
NFR-EAWVNF-002	The auto-scaling solution has a 92.5% decrease in energy consumption (a failed request rate of up to 0% and reasonable execution times of the auto-scaling process for different problem sizes).
FR-EAWVNF-004.00	One of the modules that form the auto-scaling solution is the energy-aware orchestrator, which calculates the energy consumption (using the energy consumption model detailed in Section 2.2.3.1) according to the location of the VNFs, and assigns the applications/VNFs to the most energy-efficient node.
NFR-EAWVNF-006	The energy-aware orchestrator, as well as the ENI module, includes the expected performance as a constraint, to reduce energy consumption without compromising throughput.

2.2.2 Guidelines from D2.2

The design of this solution has followed the guidelines established in D2.2 [1], Section 4.1. The computational complexity of an NI model has a strong correlation with its latency performance and energy consumption. For this reason, the solution scaling can adapt its operation to the current workload trend, being more and less resource-preserving conditioned to the changing context (current workload) and the type of system to reduce complexity. Also, the flexibility of the solution design allows the use of other prediction models, simply replacing the workload predictor component with an alternative one.

2.2.3 Solution description

Horizontal auto-scaling makes it possible to adapt the amount of infrastructure resources available to the workload by increasing/decreasing the number of nodes while optimizing energy consumption when the demand for services or the workload varies dynamically. There exist two types of energy consumption: dynamic energy consumption, which depends on the workload of nodes and idle energy consumption, which is the energy consumption of a node for remaining active [6]. This solution decreases the energy consumption at two different points: at the scheduler (when deciding the assignment of VNFs to nodes), prioritizing the allocation of computational load to energy-efficient nodes (reducing the dynamic energy consumption); and through an auto-scaling process, which puts in sleep mode those nodes that are considered non-essential (decreasing the idle energy consumption) in the next period of time.

Apart from reducing the (idle) energy consumption of the nodes themselves, non-IT appliances such as cooling and lighting can consume around 33%-52% of the total energy of an infrastructure of up to 500 nodes [10].

Most of the existing auto-scaling solutions are reactive as they scale the provision of resources using rule-based policies that apply to events (e.g., drastic resource cut down). When an event occurs, they use heuristics for the allocation and deallocation of application resources. However, as resource provisioning is done when needed, reactive auto-scaling is slower since it takes time to decide and perform, so it could not satisfy a service time response objective during a certain time [11]. In addition, for dynamic workloads that require frequent scaling up and down, the cost (in time and energy) derived from this process could be too high. Alternatively, proactive (i.e., predictive) methods can perform better in minimizing response time, satisfying service latency objectives, and reducing auto-scaling interventions, as they are able to forecast future workloads. Although the reduction of activated resources decreases the energy consumption of the infrastructure, it increases the risk of failed requests. Thus, the development of predictive auto-scaling policies under dynamic workloads involves an important challenge when they are put into practice, especially for edge-based infrastructures.

In fact, proactive auto-scaling solutions for Edge Computing systems are scarce. Due to heterogeneous computing resources, proactive auto-scaling for Edge Computing is more complex and time-consuming than in cloud systems [12] [13].

Figure 4 shows an overview of the solution (presented in [14]), which is a proactive horizontal auto-scaling framework especially well suited for edge infrastructures with different operation modes that considers the idle and dynamic energy consumption of the nodes as well as the energy consumption of node scaling. Consider an infrastructure that serves user demand for several applications (contained in a repository). The infrastructure has one (or several) master nodes, which manage several associated worker nodes to serve user requests. These master nodes run an energy-aware orchestrator, which prioritizes the allocation of VNFs to energy-efficient nodes (thus minimizing dynamic energy consumption). The system is connected to a proactive auto-scaling module, which is responsible for determining which nodes in the infrastructure should be put into sleep mode for the next time period (thus reducing the idle energy consumption). This auto-scaling module uses an ML-based predictive system (Workload predictor) to forecast the expected workload at the next time interval (e.g., five minutes), which is connected to the Essential Node Identifier (ENI) module that decides which nodes should remain active and which should be put in sleep mode (horizontal scaling). To this aim, the proactive auto-scaling module uses the information provided by the Workload predictor and the (software and hardware) information of the infrastructure, as well as the node' status.

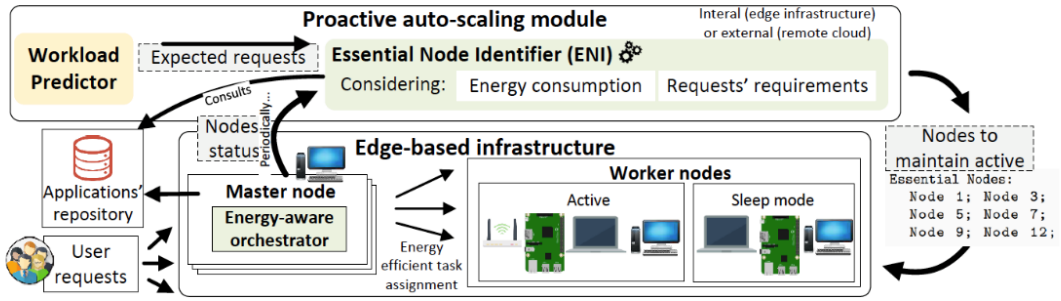

Figure 4. *Energy-aware autoscaling overview.*

The proposed auto-scaling solution has four resource reservation policies, applicable to two different operation modes (i.e., a total of eight configurations), each of which can be used with the desired VNF assignment policy. In addition, the benefit of dynamic energy saving of this solution is evaluated by introducing an energy-aware task allocation policy (D5.2). To this purpose, the auto-scaling module is connected to the open-source EdgeCloudSim edge environment simulator [15] and applied to three different workload scenarios: ascending, descending and fluctuating.

2.2.3.1 *Energy Consumption Model of the proactive auto-scaling solution*

The energy consumption model considers the two types of energy consumption previously described: dynamic (due to computation and communication) and base (by the fact of remaining nodes active) energy consumption. Within dynamic energy consumption, we distinguish the one associated with computation and communication. The computational energy consumption is influenced by several factors, such as the usage of CPU, storage, and RAM (Random Access Memory), being the CPU usage the most influential factor [13] and the one on which most Edge Computing approaches base their energy consumption models.

The formulas below contain the expressions used in this work to calculate the energy consumption (J) associated with computation ($eComp$), as well as the energy consumption for remaining active ($eIdle$), for being inactive ($eSleep$), for switching on and off nodes ($eSwitch$) [16] and for data sending and receiving ($eSend$ and $eRcpt$):

$$eComp_{n,i} = (1 - \alpha_n) eMax_n v_i \frac{w_i}{CPU_n} ew_n + eDeploy_n$$

$$eIdle_n = \alpha_n eMax_n tew_n$$

$$eSleep_n = \beta_n eMax_n tew_n$$

$$eSwitch_n = s_n eSOn_n$$

$$eSend_{dataUp_i,n} = P_n^{Tx} \frac{(1 + pR) dataUp_i}{R_n^{Tx}} ew_n$$

$$eRcpt_{dataDown_i,n} = P_n^{Rx} \frac{(1 + pR) dataDown_i}{R_n^{Rx}} ew_n$$

where $eMax_n$ is the energy consumption of node n when it is fully-utilized (in terms of CPU); $v_n \in (0,1)$ its CPU utilization ratio; w_i is the number of CPU instructions required by VNF_i(obtained on practice using tools like perf**¡Error! No se encuentra el origen de la referencia.**); t is the time (s); and α_n, whose value is between 0 and 1, represents the fraction of the idle energy consumption (e.g., 40%) of node n [7]. This model is based on the observation that the energy consumption is linear to the CPU utilization ratio, which depends on the computation load. $eDeploy_n$ is the (fixed) amount of energy (J) required by node n to create the container of the task. β_n, which is part of the expression that denotes the energy consumption of a node when sleeping, is the fraction of the sleep energy consumption of node n. Regarding switching energy consumption, s_n is 1 if the node was sleeping before the switching, 0 on the contrary, being $eSOn_n$ the energy consumption for node switching on [16]; $eSend$ and $eRcpt$ denote the energy consumption for data sending and receiving (the amounts of data (bits) $dataUp$ and $dataDown$) respectively- being pR the probability of retransmission due to packet loss. P^{Tx} and P^{Rx} represent the transmission powers, while R^{Tx} and R^{Rx} represent the data sending and receiving transmission rates. The variable ew (energy weight), presented in all the expressions, allows to select the importance of saving energy in each device separately. Thus, if the intention of the infrastructure manager is to focus the reduction of energy consumption on some specific devices or, on the contrary, to execute the greatest number of requests on others, it is sufficient to set this variable to 1 on the devices on which the energy saving is focused and to 0 on the others. This weight-based system allows to provide flexibility to the deployment solutions and eases the definition of policies according to the infrastructure necessities. An example of practical use would be to prioritize the execution of tasks on devices powered by green energy, therefore reducing the carbon footprint.

2.2.3.2 Workload predictor

The workload predictor component is responsible for forecasting the future workload that will be submitted to the edge infrastructure by the users based on historical data. This component is implemented with the Context-aware Prediction Framework presented in [11], which we integrated as part of our solution. We have chosen this framework because it uses different machine learning algorithms and applies them according to the context: Linear Regression, Support Vector Regression, and Neural Networks. The framework consists of two main components: (1) the Context Analyzer, which identifies the execution context of the application in order to select the machine learning model that suits its workload pattern (i.e., decreasing, increasing, and fluctuating workload); and (2) the Selector Algorithm, that makes the decision of selecting one of the machine learning algorithms based on the workload pattern identified by the Context Analyzer.

2.2.3.3 Essential Node Identifier

The solution includes the Essential Node Identifier (ENI) module, which is responsible for informing about the nodes that must be kept active (and which ones should be put into sleep mode) to satisfy the expected workload and minimize the (idle) energy consumption. For this purpose, it uses the information on the expected workload received from the workload predictor and the infrastructure's current status (see *Figure 4*). Then, using the energy model presented above, it evaluates the allocation of resources needed to meet the expected workload. This module has two different operation modes (OM), which relate to being more and less resource-preserving. In the first one (OM1), the ENI module can order the deactivation (putting in sleep mode) of nodes that are executing a task at the time when the auto-scaling process is being performed if deemed appropriate. In this case, the deactivation would be performed at the time the node finishes the execution of the assigned tasks. While this is a priori the most energy-efficient solution (as just the needed resources are active), it introduces some delay as the nodes require time to become active after being deactivated. Therefore, it is possible to receive requests demanding resources that are being activated, resulting in failed requests. A second and more resource-preserving operation mode (OM2) is defined to maximize the number of successful requests. Unlike OM1, OM2 keeps active the nodes that are running an application at the time the auto-scaling is performed, thus maintaining more active nodes.

Module 1 Essential Node Identifier (ENI)

Data: Nodes; currentStatus; expectedWorkload (expWl); autoscalingInterval; operationMode (OM1/OM2)
Result: actN

1 Nodes ← currentStatus
2 neededResources = getAssociatedResources (expWl)
// Parameters to optimize:
3 activeNodes (actN); computationEnergyConsumption (compEC); transmissionEnergyConsumption (sendEC); receptionEnergyConsumption (rcptEC); idleEnergyConsumption (idleEC); switchingEnergyConsumption (switchEC); expectedWorkload (expWl)
// Acronyms: freeResources (freeR); dataToUpload (dataUp); dataToDownload (dataDown)
// Constrains:
4 sol = Optimize()
5 sol.addConst (aN ⊂ Nodes)
6 sol.addConst(activeResources == $\sum_{n \in aN} n_{freeR}$)
7 sol.addConst(neededResources ≤ activeResources)
8 **if** *operationMode* == *OM2* **then**
9 | sol.addConst($\forall n \in actN$: **if** (n.isworking) **then**: n ∈ actN)
10 **end**
11 sol.addConst(compEC == $\sum_{n \in aN}$ eComp(n,expWl)
12 sol.addConst(sendEC == $\sum_{n \in actN}$ eSend(n_{RTx}, n_{PTx},avg($expWl_{dataUp}$)
13 sol.addConst(rcptEC == $\sum_{n \in actN}$ eRcpt(n_{RRx}, n_{PRx}, n_{freeR},avg($expWl_{dataDown}$)
14 sol.addConst(idleEC == $\sum_{n \in actN}$ eIdle(n_{eIdle}, autoscalingInterval)
15 sol.addConst(sleepEC == $\sum_{n \in (N \not\subset actN)}$ eSleep(n_{eSleep}, autoscalingInterval)
16 sol.addConst(switchEC == $\sum_{n \in N}$ eSwitch(n_{eSOn}, n.isworking, $n \in actN$))
17 sol.minimize(compEC + sendEC + rcptEC + idleEC + sleepEC + switchEC);
18 satisfiable = sol.checkSatisfability() // Checking the satisfacibility
19 **if** *satisfiable* **then**
20 | result = sol.solve()
21 **end**
22 **return(result.actN)**

Figure 5. *Essential node identifier algorithm.*

The ENI's behavior (the ENI algorithm is shown in Figure 5) is posed as a constraint satisfaction problem. First, the module updates the nodes' information with the current status and gets the needed resources from the expected workload received as input. The constraints that form the problem do the following: assure that the set aN is a subset of $Nodes$ (line 5); line 6 includes the resources of the active nodes in the $activeResources$ set, while line 7 checks that the resources included in $activeResources$ are enough to meet the expected workload; in case of the OM2, lines 8 to 10 include in the solution the nodes that are currently executing any other delegated task; line 11 calculates the computation energy consumption, while lines 12-13 estimate the communication energy consumption (sending and receiving respectively); line 14 calculates the idle energy consumption (using the auto-scaling interval received as an input); line 15 estimates the sleep energy consumption, while line 16 calculates the switching energy consumption; line 17 minimizes the energy consumption; line 18 checks the problem satisfiability, while lines 19-21 ask for the solution if so. Finally, line 22 returns the solution.

An experimental evaluation of this solution will be reported in D5.2.

2.2.4 Mapping to major standards (if applicable) and gaps identification

The main benefits of using this predictive auto-scaling framework are: (i) it has been shown that the Workload predictor obtains predictions at least 10% more accurately than others, and (ii) it adapts to the dynamic workloads.

The ENI module is solved using Z3, an SMT solver. Thus, (1) the algorithm always returns a solution (unlike heuristic-based algorithms), which guarantees that the deployment is feasible or the lack of infrastructure's resources if no solution is found; and (2) a larger number of constraints (required to solve the problem at hand) helps SMT-solvers to reduce the search space and to find the solution faster. Unlike solvers that use Integer Linear Programming or Mixed Integer Linear/Nonlinear Programming, SMT solvers neither restrict the variable types involved in the problem formulation nor the degree of the equations

[17]. Additionally, it is proved that Z3, and concretely its optimization module (νZ), returns optimal solutions. Nevertheless, the flexibility of our approach may allow the use of any other mathematical model for optimization.

2.2.5 N-MAPE-K representation

The following table contains the decomposition of the proactive auto-scaling solution into the different components of the MAPE-K.

***Table 8**. N-MAPE-K loop mapping of proactive autoscaling solution and VNF placement.*

N-MAPE-K module	Solution module
Monitor	The current workload and the status of the nodes in the infrastructure (CPU, RAM, disk and bandwidth resources) are monitored periodically.
Analyze	Analyze the current workload to identify a workload pattern and select the prediction algorithm to provide the expected workload for the next time interval with a minimal number of service fails.
Plan	Decide the optimal horizontal autoscaling (which nodes should keep active to serve user requests) in terms of energy consumption at the next time interval according to the predicted workload (Workload predictor) and the current status of the infrastructure.
Execution	Run the autoscaling process (activate/deactivate nodes) using the operation mode that maximizes the number of successful requests.
knowledge	Workload prediction model (ML algorithms), Energy Model, Infrastructure Status, Infrastructure information (software and software), Workload (current and expected), applications requirements (CPU, RAM, disk and BW resources)
Training Loss / State – Actions – Rewards	The ENI module has several modes of operation, more and less resource-preserving. Depending on the failed requests that the system gets over time, it is possible to make the system change its operating mode to adjust to the needs of the infrastructure. For this purpose, the percentage of failed requests would be compared with the desired value.

2.3 Energy-driven vRAN orchestration

Energy consumption at the network edge, namely at the base stations, is one of the most urgent issues in future mobile networks. At the same time, the increasing softwarization and virtualization of these nodes, the base stations, make their configuration and the allocation of the resources involved in their operation an intricate task. This thread tackles these two issues jointly by proposing a zero-touch data-driven method for orchestrating the operation of virtualized base stations while controlling their energy consumption and trading-off energy costs with performance. The gist of our approach is a Bayesian learning technique that allows runtime adaptation of the scheduling policies based on the observed network conditions, user loads (demands), and system performance.

2.3.1 Requirements in D2.2

The mapping between the requirements established for energy-driven vRAN orchestration in D2.2 [1] is presented in the following table.

***Table 9.** Energy-driven vRAN orchestration. Requirement satisfaction.*

Requirement from D2.2 [1]	Rationale
FR-EAWVNF-000	This NI Functionality monitors the power consumption profile of O-RAN base stations.
FR-EAWVNF-005	This NI Functionality uses the power consumption profile of O-RAN base stations to optimize their operation.
NFR-EAWVNF-004	The BP-vRAN algorithm aims at balancing network throughput and energy consumption in vRANs.
NFR-EAWVNF-005	The SBP-vRAN algorithm aims at maximizing network throughput given power consumption constraints in vRANs.

2.3.2 Guidelines from D2.2

The design of this solution has followed the guidelines established in D2.2 [1], Section 4.1, to tailor Bayesian learning techniques to a low inference time and low energy consumption NI. In detail, as Bayesian learning uses Gaussian Processes (GPs) to model the latent relation between context, actions, and reward, the number of computations during inference time is minimal compared to AI models that use vast and complex neural networks. This small number of operations during inference time makes Bayesian learning a technique that consumes very few energy resources.

2.3.3 Solution description

In D4.1 [3], Section 3.2, we described a problem that exploiting O-RAN's architecture that is integrated into DAEMON optimizes the operation of O-RAN base stations in terms of network performance and energy consumption through non-real-time policies. We formulated two different problems using a Contextual Bandit (CB) model. The first problem (Case 1) aimed at balancing network performance and cost (power consumption) and was labeled as BP-vRAN. The second problem (Case 2) aimed at maximizing network performance subject to hard power constraints and was labeled as SBP-vRAN.

Many algorithms for solving contextual bandit problems assume that there is a feature vector associated with each action, and the objective function is linear in that vector. However, this assumption does not hold here for the following reasons. Firstly, our objective function is not linear, as we can observe in D4.1 [3], Section 3.2.2.1. Secondly, the function values associated with different actions (i.e., vBS control policies) are correlated. Intuitively, we can think that a small change in some configuration parameter (e.g., airtime) will induce a small change in the vBS consumed power. This is actually evaluated experimentally in Figure 3.2.7, in D4.1 [3]. This means that we can obtain information about unobserved context-control pairs by observing nearby actions, thus reducing the exploration time. Based on these observations, we propose a Bayesian optimization method where we model the objective function as a sample from a Gaussian Process (GP) over the joint context-control space. This non-parametric estimator captures the aforementioned non-linearities and correlations and provides predictive uncertainty on the function estimation. Hence, effectively addresses the exploration vs. exploitation trade-off.

Function estimator. We use a GP as a function estimator, which is a collection of random variables following joint Gaussian distributions. Let $z \in \mathcal{Z} = \Omega \times \mathcal{X}$ denote a context-control pair. We model the unknown objective function -as described in Section 3.2.21 of D4.1 [3]- as a sample from a $GP\big(\mu(z), k(z,z')\big)$, where $\mu(z)$ is its mean function and $k(z,z')$ is its covariance function or kernel. Without loss of generality, we assume $\mu := 0$, which we refer to as the *prior distribution*, not conditioned on data, and with bounded variance $k(z,z) < 1$. Given this prior and a set of observations, the mean and covariance of the *posterior distribution* can be computed using closed-form formulas. Let $y_T = [u_1, \dots, u_T]$ be a vector of noisy samples (assuming *i.i.d.* Gaussian noise $\sim N(0, \sigma^2)$) at points $Z_T = [z_1, \dots, z_T]$. Then, the posterior distribution of the objective function follows a GP distribution with mean $\mu_T(z)$ and covariance $k_T(z,z')$:

$$\mu_T(z) = k_T(z^\top)(K_T + \zeta^2 I_T)^{-1} y_T$$
$$k_T(z,z') = k(z,z') - k_T(z^\top)(K_T + \zeta^2 I_T)^{-1} k_T(z')$$

where $k_T(z) = [k(z_1,z), \dots, k(z_T,z)]^\top$, $K_T(z)$ is a kernel matrix defined as $[k(z,z')]_{z,z' \in Z_T}$, and I_T is the T-dimension identity matrix. These equations allow us to estimate the distribution of unobserved values of $z \in \mathcal{Z}$ based on the prior distribution, the vector Z_T, and the function observations y_T.

Kernel function. The selection of the kernel is crucial. Because it shapes the prior and posterior Gaussian Process (GP) distributions by encoding the correlation between the values of the objective function of every pair of points. Namely, $k(z,z')$ indicates the similarity between $u_t(z)$ and $u_t(z')$. In other words, the kernel characterizes the smoothness of the function. Based on our analysis in **¡Error! No se encuentra el origen de la referencia.**, we selected the kernel for this problem satisfying two properties: *stationarity* and *anisotropicity*. On the one hand, a kernel $k(z,z')$ is stationary if it depends only on the distance between z and z', which means that it is invariant to translations in $\mathcal{Z}$. On the other hand, a kernel is anisotropic when the encoded smoothness is different among the different dimensions of $\mathcal{Z}$. That means that an anisotropic kernel is not invariant to rotations in $\mathcal{Z}$. We encode the smoothness of the objective function u with a length-scale vector $\mathcal{L} = [l_1, \dots, l_N]$, where N indicates the number of dimensions of $\mathcal{Z}$. Thus, the distance between two points based on the length-scale vector can be written as

$$d(z,z') = \sqrt{(z-z')^\top L^{-2} (z-z')},$$

where $L = \mathrm{diag}(\mathcal{L})$ is a diagonal matrix of the length-scale values. Hence, we selected the anisotropic version of the Matérn kernel, which satisfies the properties discussed above. In detail, we select the anisotropic version of the Matérn kernel. To this end, following standard practice, we particularize the Matérn kernel with parameter $\nu = \frac{3}{2}$ to devise a simple expression that guarantees that the objective function is at least once differentiable, which yields:

$$k(z,z') = \left(1+\sqrt{3}d(z,z')\right)\exp\left(-\sqrt{3}d(z,z')\right).$$

To improve performance, we can optimize the hyperparameters $\mathcal{L}$ and the noise variance ζ^2 before running the algorithm by maximizing the likelihood estimation over prior data and we keep these values constant over time.

Acquisition function. The acquisition function selects one control x_t at each period t based on the posterior distribution of the objective function over the context-control pairs. To this aim, we use the Upper Confidence Bound (UCB) method:

$$x_t = \underset{x\in\mathcal{X}}{\operatorname{argmax}}\ \mu_{t-1}(\omega_t, x) + \sqrt{\beta_t}\sigma_{t-1}(\omega_t, x).$$

Where ω_t is the observed context at time t, β_t is a weighting parameter and $\sigma_t(z) = k_t(z,z)$.

In what follows, we describe the solution to both problems founded in Bayesian learning.

2.3.3.1 BP-vRAN: Balancing network performance and power consumption

We formalize our approach to solving the first CB problem using Bayesian learning, which we refer to as BP-vRAN (Bayesian optimization for Power consumption in vRANs), in Figure 6. At the beginning of each decision period t, a context ω_t is observed (line 3). Then control x_t is decided based on the GP posterior and the acquisition function (line 5). At the end of t, the throughput and consumed power are observed and the value of the objective function is computed (lines 6-7). The new measurements are included in Z_t and y_t to improve the posterior distribution in $t+1$ (lines 8-10).

The choice of a value for β_t is very important since it controls the trade-off between exploration and exploitation. Larger values of β_t lead the acquisition function to select controls with higher uncertainty while, conversely, controls already known to be high-performing (though not necessarily *highest-performing*) are selected when β_t takes smaller values. Following conventional approaches, we select

$$\beta_t = 2B^2 + 300\gamma_t \ln^3\left(\frac{t}{\epsilon}\right)$$

where $\epsilon \in (0,1)$, $B \geq \| u \|_k$ is an upper bound on the Reproductive Kernel Hilbert Space (RKHS) norm of u, and γ_t is the maximum mutual information gain obtained from u after t observations.

Algorithm 1 BP-vRAN: Performance and cost balancing

1: **Inputs:** Control Space $\mathcal{X}$, kernel k, β
2: **Initialize:** $y_0 = \emptyset$, $Z_0 = \emptyset$
3: **for** $t = 1, \ldots, T$ **do**
4: Observe the context ω_t
5: $x_t = \text{argmax}_{x\in\mathcal{X}}\ \mu_{t-1}(\omega_t, x) + \sqrt{\beta_t}\sigma_{t-1}(\omega_t, x)$
6: Measure $R_t^{dl}(\omega_t^{dl}, x_t^{dl})$, $R_t^{ul}(\omega_t^{ul}, x_t^{ul})$ and $P_t(\omega_t, x_t)$ at the end of the decision period t
7: Compute $u_t(\omega_t, x_t)$ using (1), (2) and (3)
8: Update $Z_t \leftarrow Z_{t-1} \cup [\omega_t, x_t]$
9: Update $y_t \leftarrow y_{t-1} \cup u_t(\omega_t, x_t)$
10: Perform Bayesian update to obtain μ_t and σ_t
11: **end for**

Figure 6. *BP-vRAN algorithm.*

where equations (1), (2) and (3) refer to:

$$r(\omega_t, x_t) \coloneqq \log\left(1+\frac{R^{dl}(\omega_t^{dl}, x_t^{dl})}{d_t^{dl}}\right) + \log\left(1+\frac{R^{ul}(\omega_t^{ul}, x_t^{ul})}{d_t^{ul}}\right)$$

$$u(\omega_t, x_t) \coloneqq r(\omega_t, x_t) - \delta B\big(P(\omega_t, x_t)\big),$$

and

$$B(k) \coloneqq \frac{1+e^{ab}}{e^{ab}}\left(\frac{1}{1+e^{-a(k-b)}} - \frac{1}{1+e^{ab}}\right).$$

respectively, as defined in D4.1 [3], Section 3.2.2.

2.3.3.2 SBP-vRAN: Safe Bayesian optimization

Imposing hard constraints, as proposed in D4.1 [3], Section 3.2.2.2, compounds the problem. Prior works, e.g., in robotics and other areas, have proposed Bayesian optimization algorithms with *safety constraints*. Their main idea lays upon the definition: every t we define a subset of *safe* controls $S_t \subseteq \mathcal{X}$ that satisfy the

constraints with certainty. Then, they implement an exploration process that expands the safe set while seeking a safe action with high performance. Unfortunately, these works do not consider contextual information, which clearly affects the safe set, i.e., $S_t(\omega_t) \subseteq \mathcal{X}$. To the best of our knowledge, only SafeOpt [18] proposes a contextual safe learning algorithm. However, although this algorithm provides theoretical guarantees, its acquisition function selects the control with the highest uncertainty among all candidates that can expand the safe set and also the potential maximizers. We found in our experiments that this approach has overly slow convergence. This practical issue has been reported in other works as well. Hence, we improve this methodology by employing the acquisition function of CGP-UCB but *constrained to the safe set*.

We denote $y_T^f = [r_1, \ldots, r_T]$ the vector of reward samples at T and $y_T^c = [P_1, \ldots, P_T]$ the power consumption samples. We use one GP for the reward and one for the power constraint. Both GPs have the same prior distribution and kernel but different hyperparameters. The posterior distribution can be computed using the expressions presented in D4.1 [3], Section 3.2.2.2, and replacing y_T by y_T^f or y_T^c, for each GP. We denote the posterior mean and covariance of the reward at T as $\mu_T^f(z)$ and $k_T^f(z, z')$, and $\mu_T^c(z)$ and $k_T^c(z, z')$ for the power, respectively. The initial safe set $S_0 \subseteq \mathcal{X}$ is common for all contexts and includes low power consumption configurations (vBS close to idle). This is worst-case S_0 can be expanded using prior data. At each period, S_t is computed based on the posterior distribution of the power consumption provided by the GP. We assume the true value of the power consumption at time t is within the interval $[\mu_t^c(z) \pm \beta_t \sigma_t^c(z)]$, where $\sigma_t^c(z) = k_t^c(z, z)$. Using the posterior distribution, we define the safe set a time t and for a given context ω_t as:

$$S_t = \{x \in \mathcal{X} \mid \mu_{t-1}^c(\omega_t, x) + \beta_t \sigma_{t-1}^c \omega_t, x) \leq P_{\max}\}.$$

The controls are selected at each decision period t using the CGP-UCB policy constrained to safe set:

$$x_t = \underset{x \in S_t}{\operatorname{argmax}}\ \mu_{t-1}^f(\omega_t, x) + \sqrt{\beta_t}\sigma_{t-1}^f(\omega_t, x),$$

where $\sigma_t^f(z) = k_t^f(z, z)$.

Algorithm 2 SBP-vRAN: Safe online optimization

Inputs: Control Space $\mathcal{X}$, Initial safe set S_0, kernel k, β
Initialize: $y_0^f = \emptyset$, $y_0^c = \emptyset$, $Z_0 = \emptyset$
for $t = 1, \ldots, T$ **do**
  Observe the context ω_t
  $S_t = S_0 \cup \{x \in \mathcal{X} \mid \mu_{t-1}^c(\omega_t, x) + \beta_t \sigma_{t-1}^c(\omega_t, x) \leq P_{\max}\}$
  $x_t = \operatorname{argmax}_{x \in S_t}\ \mu_{t-1}(\omega_t, x) + \sqrt{\beta_t}\sigma_{t-1}(\omega_t, x)$
  Measure $R_t^{dl}(\omega_t^{dl}, x_t^{dl})$, $R_t^{ul}(\omega_t^{ul}, x_t^{ul})$ and $P_t(\omega_t, x_t)$ at the end of the decision period t
  Compute $r_t(\omega_t, x_t)$ using (1)
  Update $Z_t \leftarrow Z_{t-1} \cup [\omega_t, x_t]$
  Update $y_t^f \leftarrow y_{t-1}^f \cup r_t(\omega_t, x_t)$
  Update $y_t^c \leftarrow y_{t-1}^c \cup P_t(\omega_t, x_t)$
  Perform Bayesian update to obtain μ_t^f, σ_t^f, μ_t^c and σ_t^c
end for

***Figure 7**. SBP-vRAN algorithm.*

We summarize our approach, named SBP-vRAN (Safe Bayesian optimization for Power consumption in vRANs), in Figure 7. Note that it does not explicitly expand the safe set like it is done in SafeOpt. Instead, a new observation is included at each t, which updates the posterior distribution of the power consumption and therefore alters the safe set indirectly. In our preliminary experimental campaign in D4.1 [3], Section 3.2.1, we observed that, roughly speaking, controls with higher throughput are associated with higher power consumption. This leads our algorithm to explore controls *at the boundary of the selected constraint*. As a result, the uncertainty around their neighborhood decreases, which allows the algorithm to include more controls in the safe set. That is, SBP-vRAN's acquisition function exploits the structure of our problem to effectively expand the safe set, as our experimental evaluation presented in the next section demonstrates.

An experimental evaluation of this solution will be reported in D5.2.

2.3.4 Mapping to major standards (if applicable) and gaps identification

Both of the learning algorithms, BP – vRAN and SBP – vRAN, can be implemented as O-RAN xApps, within O-RAN's Non-Real-Time RIC.

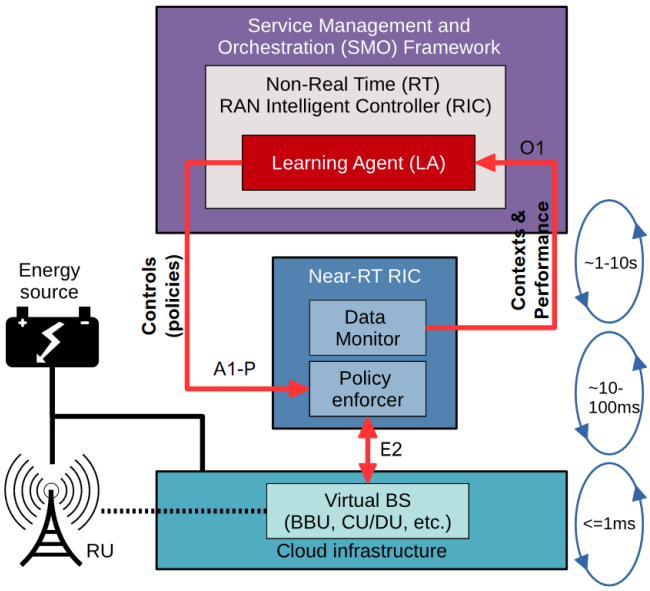

Figure 8. *Integration of energy-driven vRAN orchestration (network intelligence) functionality in O-RAN architecture.*

As shown in **¡Error! No se encuentra el origen de la referencia.**, a learning agent or LA (BP – vRAN or SBP – vRAN) is in charge of orchestrating the resources assigned to the network slice via a set of control policies. That is, the LA runs in the Service Management and Orchestration (SMO) in the timescale of seconds. Radio policies are enforced by the near-RT RIC by using O-RAN's A1 and E2 interfaces, as shown in the figure. Feedback from the data-plane components is collected by the non-RT RIC (via O-RAN's O1 interface) and then fed to the LA. This allows us to use Reinforcement Learning (RL) to solve our problem.

2.3.5 N-MAPE-K representation

Following WP2 guidelines, in the following, we outline the N-MAPE-K representation of this functionality:

Table 10. *N-MAPE-K loop mapping of energy-driven vRAN orchestration NI.*

Analytics	**Description**	
	BP – vRAN	SBP – vRAN
Sensors + Monitor	Channel conditions: SNR measurements, Traffic demands: Buffer State Reports and Downlink buffer occupation.	
Analyze	Inputs from downlink and uplink are concatenated into a single context that can serve as the input to the NI function.	
Plan	A Bayesian learning model uses Gaussian processes in conjunction to a Matern kernel to learn the correct balance between network performance and energy cost. The model outputs three actions: MCS policy, airtime policy, TX power policy.	A Bayesian learning model uses Gaussian processes in conjunction to a Matern kernel to learn the correct balance between network performance and energy cost. The model uses safe constrains to improve the results of the exploration phase. The model outputs three actions: MCS policy, airtime policy, TX power policy.
Execution + Effector	Three APIs exposed by the base station (O-RAN E2 interface).	
Knowledge	A model of the network performance and energy cost for different configurations of MCSs, transmission power and airtime.	
Training Loss / State – Actions – Rewards	• States: Latent representation of the input context data • Actions: configuration of the MCSs, transmission power in the DL and airtime • Rewards: Energy cost minimization	

2.4 2Energy-driven joint vRAN & edge service orchestration

The solution to this problem, described in D4.1 [3], Section 3.3, will be provided in D4.3.

3 Capacity Forecasting

Capacity forecasting is a task in anticipatory networking that aims at reserving the resources needed to meet the upcoming traffic demand. The concept is intendedly general and may apply to different network domains or entities, as well as to diverse definitions of demand, such as aggregates or service-level ones. In the following, we present four different use cases in the context of capacity forecasting, i.e., anticipatory allocation of resources, VM reservation in a network core datacenter, minimization of video streaming slice OPEX, and prediction-assisted network scheduling.

3.1 Anticipatory capacity allocation with hybrid statistical-learning models

This capacity forecasting use-case targets a scenario where anticipatory NI is in charge of capacity allocation to slices. Here, as in other capacity forecasting contexts, sheer accuracy is not the most relevant metric. Instead, it is critical that the capacity prediction stays above the actual load with a very high probability because underestimation determines the allocation of insufficient capacity to slices, hence causing service disruption on the user side. Under-provisioning also triggers violations of the SLA between the slice tenant and the network operator, which thus incurs substantial economic penalties. Clearly, this must be avoided without allocating exceedingly large amounts of unnecessary resources that have a cost for the operator. We refer the reader to Section 4.1 of D4.1 [3] for a complete definition of the problem.

3.1.1 Requirements in D2.2

The mapping between the requirements established for capacity forecasting solutions in D2.2 [1] and the proposed model is presented in the following table.

Table 11. *Anticipatory capacity allocation solution. Requirement satisfaction.*

Requirement from D2.2 [1]	Rationale
FR-CFORE-000	This NI functionality addresses one specific problem in the context of capacity forecasting, hence contributes to meeting the requirement of designing models for capacity prediction that support Network Intelligence (NI) algorithms across the mobile network architecture.
FR-CFORE-002	This NI functionality leverages a dedicated loss function that accounts for the monetary costs incurred by the operator, hence embeds a training of the model that produces practical predictions targeted at resource management.

3.1.2 Guidelines from D2.2

This solution for this NI functionality abides by the guidelines established in Section 4.1 of D2.2 [1] of the DAEMON project in terms of adaptation of legacy AI models to the specificities of real-world NI problems. Specifically, the model employed in this task uses a customized loss function that is carefully developed based on expert system knowledge and accounts for the actual monetary costs incurred by the network operator and associated with the capacity forecast output. In this way, the proposed solution avoids the common loss-metric mismatch in full alignment with the strategy outlined in Section 4.1.2 of D2.2 [1].

In addition, the solution also follows the guidelines established in Section 4.2 of D2.2 [1], by investigating alternative methods that allow for broadening the spectrum of learning and optimization tools for practical NI tasks. Indeed, the model employed to perform the anticipatory capacity allocation adheres to the full hybrid approach that is proposed in that Section 4.2.4 of D2.2 [1], as a better alternative to pure deep learning AI models commonly used in the literature for network traffic forecasting.

3.1.3 Solution description

To address this problem, we designed a NI algorithm named Thresholded Exponential Smoothing with Recurrent Neural Network, or TES-RNN [19], a time series forecasting model which combines a classical Exponential Smoothing (ES) statistical model with a Recurrent Neural Network (RNN) architecture. A complete description of the TES-RNN model is provided in Section 4.2.4 of D2.2 [1]of the DAEMON project.

To solve the target anticipatory capacity allocation problem, the TES-RNN is trained with a custom loss function *F* that accounts for the monetary costs incurred by the operator and is determined by the output capacity prediction. The definition of the loss is presented in Section 4.1.1 of D4.1 [3].

3.1.4 Mapping to major standards (if applicable) and gaps identification

This is a generic NI algorithm that could be deployed at different network equipment or datacenters. It is thus a module that could be implemented as a xApp/rApp in O-RAN or NWDAF in 3GPP. Therefore, the solution closes an algorithmic gap in terms of NI and can integrate into relevant standards.

3.1.5 N-MAPE-K representation

The following table contains the decomposition of the anticipatory capacity forecasting solution into the different components of the N-MAPE-K representation.

Table 12. *Anticipatory capacity allocation solution. N-MAPE-K mapping.*

N-MAPE-K module	Solution module
Monitor	The current aggregate traffic demands, e.g., measured in Mbps, served by the target network equipment or datacenter where the capacity allocation must be performed.
Analyze	Current and historical aggregate traffic demands are concatenated to produce a time series of demand values at the target network equipment or datacenter.
Plan	The NI algorithm is a TES-RNN model trained with a custom loss function and outputs the amount of resources needed to accommodate the future traffic demands at the target network equipment or datacenter over a given horizon.
Execution	Allocate resources (e.g., compute, transport, capacity, depending on the target network equipment or datacenter) based on the forecast.
Knowledge	A substantial history (e.g., over several continuous months) of aggregate traffic demands at the target network equipment or datacenter.
Training Loss / State – Actions – Rewards	The NI algorithm is a pre-trained supervised model using a custom loss over labeled data stored in the knowledge.

3.2 Virtual Machine reservation under meta-learned loss

In this capacity forecasting use case, we consider a core network datacenter setting, where we assume that different video streaming services are assigned a dedicated Network Slice Subnet Instance (NSSI). This use case implies that the VIM responsible for controlling the datacenter resources must predict the number of VMs that need to be allocated in advance to each NSSI, so as to run the VNFs required to serve the demand generated by the corresponding mobile service. Clearly, every VM has an operating cost (e.g., due to power consumption), so it is desirable that only the strictly necessary set of VMs is reserved for each NSSI. We refer the reader to Section 4.2 of D4.1 [3] for a complete definition of the problem.

3.2.1 Requirements in D2.2

The mapping between the requirements established for capacity forecasting solutions in D2.2 [1] and the proposed model is presented in the following table.

Table 13*. Virtual Machine reservation. Requirement satisfaction.*

Requirement from D2.2 [1]	Rationale
FR-CFORE-000	This NI Functionality addresses one specific problem in the context of capacity forecasting, hence contributes to meeting the requirement of designing models for capacity prediction that support Network Intelligence (NI) algorithms across the mobile network architecture.
FR-CFORE-002	This NI functionality meta-learns a loss function that accounts for the monetary costs incurred by the operator, hence embeds a training of the model that produces practical predictions targeted at resource management.
FR-CFORE-005	This NI functionality learns autonomously the objective/loss function, and does not rely on legacy or expert-designed losses.

3.2.2 Guidelines from D2.2

This solution for this NI functionality abides by the guidelines established in Section 4.1 of D2.2 [1] of the DAEMON project in terms of adaptation of legacy AI models to the specificities of real-world NI problems. Specifically, the model employed in this task adopts the LossLeaP architecture proposed in the project to solve a capacity forecasting problem where the relationship between the prediction and the performance metric to be optimized is not known a priori by the network operator. In this way, the proposed solution avoids the common loss-metric mismatch in a fully automated way, in alignment with the strategy outlined in Section 4.1.3 of D2.2 [1].

3.2.3 Solution description

In this case, knowledge of the actual operating costs of VMs at the datacenter and information about the local VM management strategies are not directly or easily available to the operator, which prevents defining a custom loss function as done in the use case presented in Section 3.1 above. Therefore, we address the problem via the LossLeaP architecture [20], which allows meta-learning of the loss function that best suits the network management objective and uses it to train the forecasting model. A complete description of the LossLeaP model is provided in Section 4.1.3 of D2.2 [1].

3.2.4 Mapping to major standards (if applicable) and gaps identification

This is a generic NI algorithm that could be deployed at different network equipment or datacenters. It is thus a module that could be implemented as a xApp/rApp in O-RAN or NWDAF in 3GPP. Therefore, the solution closes an algorithmic gap in terms of NI and can integrate into relevant standards.

3.2.5 N-MAPE-K representation

The following table contains the decomposition of the anticipatory capacity forecasting solution into the different components of the N-MAPE-K representation.

Table 14. *Virtual Machine reservation. N-MAPE-K mapping.*

N-MAPE-K module	Solution module
Monitor	The current traffic demands, e.g., measured in Mbps, generated by the target NSSI at the datacenter where the VM reservation must be performed.
Analyze	Current and historical NSSI traffic demands are concatenated to produce a time series of demand values at the target network datacenter.
Plan	The NI algorithm is a LossLeap model trained on system observations, and outputs the number of VMs needed to accommodate the future traffic demands of the NSSI at the target network datacenter during the next VM configuration period.
Execution	Scale VMs based on the forecast. This can be realized, e.g., in O-RAN by controlling the gNB-CU-UP Capacity Information Element (IE) via the E2 interface to a CU-type E2 node.
Knowledge	The amount of NSSI demand that has not been served at the target datacenter due to insufficient VM allocation, and the utilization of the allocated VMs.
Training Loss / State – Actions – Rewards	The NI algorithm is an on-line supervised model that uses real-time observations of the knowledge to compute the monetary cost for the operator determined by the anticipatory VM allocation decision. Details on the calculation of the cost from the knowledge are in See Section 4.2.1 of D4.1 [3]. Such cost is then used as ground truth to meta-learning the loss and (in cascade according to the LossLeaP model) the actual predictor.

3.3 Minimization of video streaming slice OPEX with meta-learned loss

In this capacity forecasting setting, we target a mobile Edge environment where a set of computational facilities, each serving between 70 and 130 BSs, are located close to the radio access. We assume again that different video streaming services with dedicated Network Slice Subnet Instance (NSSI) run at the Edge facilities. Here, the management intent aims at minimizing the monetary OPerating EXpenses (OPEX) associated with running the video streaming slices at the network Edge. This maps to an objective of periodically and preemptively rescaling the compute resources assigned to each NSSI in a facility, to smoothly run the needed VNFs. We refer the reader to Section 4.3 of D4.1 [3] for a complete definition of the problem.

3.3.1 Requirements in D2.2

The mapping between the requirements established for capacity forecasting solutions in D2.2 [1] and the proposed model is presented in the following table.

Table 15. *Minimization of video streaming slice OPEX. Requirement satisfaction.*

Requirement from D2.2 [1]	Rationale
FR-CFORE-000	This NI Functionality addresses one specific problem in the context of capacity forecasting, hence contributes to meeting the requirement of designing models for capacity prediction that support Network Intelligence (NI) algorithms across the mobile network architecture.

FR-CFORE-002	This NI functionality meta-learns a loss function that accounts for the monetary costs incurred by the operator, hence embeds a training of the model that produces practical predictions targeted at resource management.
FR-CFORE-005	This NI functionality learns autonomously the objective/loss function, and does not rely on legacy or expert-designed losses.

3.3.2 Guidelines from D2.2

This solution for this NI functionality abides by the guidelines established in Section 4.1 of D2.2 [1] of the DAEMON project, in terms of adaptation of legacy AI models to the specificities of real-world NI problems. Specifically, the model employed in this task adopts the LossLeaP architecture proposed in the project to solve a capacity forecasting problem where the relationship between the prediction and the performance metric to be optimized is not known a priori by the network operator. In this way, the proposed solution avoids the common loss-metric mismatch in a fully automated way, in alignment with the strategy outlined in Section 4.1.3 of D2.2 [1].

3.3.3 Solution description

Similarly to the previous use-case presented in Section 3.2, the operator does not have apriori knowledge of the relationship between allocated resources to each NSSI and the OPEX performance. Therefore, we resort again to the DAEMON-developed LossLeaP framework [20] to design an NI algorithm that can meta-learn the correct loss from the experience of the real system response and suitably train the predictor to generate accurate forecasts of the resource allocations to NSSIs. A complete description of the LossLeaP model is provided in Section 4.1.3 of D2.2 [1].

3.3.4 Mapping to major standards (if applicable) and gaps identification

This is a generic NI algorithm that could be deployed at different network equipment or datacenters. It is thus a module that could be implemented as a xApp/rApp in O-RAN or NWDAF in 3GPP. Therefore, the solution closes an algorithmic gap in terms of NI, and can integrate into relevant standards.

3.3.5 N-MAPE-K representation

The following table contains the decomposition of the anticipatory capacity forecasting solution into the different components of the N-MAPE-K representation.

Table 16. *Minimization of video streaming slice OPEX. N-MAPE-K mapping.*

N-MAPE-K module	Solution module
Monitor	The current traffic demands, e.g., measured in Mbps, generated by the target NSSI at the datacenter where the resource reservation must be performed.
Analyze	Current and historical NSSI traffic demands are concatenated to produce a time series of demand values at the target network datacenter.
Plan	The NI algorithm is a LossLeap model trained on system observations, and outputs the amount of resources needed to accommodate the future traffic demands of the NSSI at the target datacenter during the next orchestration period.
Execution	Allocate resources (e.g., compute) based on the forecast. This can be realized, e.g., in O-RAN by controlling the gNB-CU-UP Capacity Information Element (IE) via the E2 interface to a CU-type E2 node.
Knowledge	The economic OPEX cost of the allocated capacity and the monetary penalty induced by the selected allocation, as computed by the service provider based on QoE/MOS information of the end users and according to the NSSI service-level agreement (SLA).
Training Loss / State – Actions – Rewards	The NI algorithm is an on-line supervised model that uses real-time observations of the knowledge to compute the monetary cost for the operator determined by the anticipatory resource allocation decision. Details on the calculation of the cost from the knowledge are in Section 4.3.1 of D4.1 [3]. Such cost is then used as ground truth to meta-learning the loss and (in cascade according to the LossLeaP model) the actual predictor.

3.4 Learning with Predictions for Edge Caching and Computing

This section describes our solution for online learning-based network management using forecasting models (or predictions). Our end goal, as was identified in D4.1 [3] is to propose and evaluate a hybrid learning model that combines offline and online training in order to achieve adaptive learning

performance (online learning) while leveraging any priors available (offline learning), whenever these are accurate. This approach follows the project's rationale of revisiting fundamental ideas on AI/ML approaches in order to tailor them to the needs of emerging communication systems. And this thread does exactly this: it bridges the gap between deep learning-based models that rely solely on offline training (which admittedly can be often inaccurate) and online learning models that rely solely on streaming data and ignore offline models. We prove mathematically and demonstrate through data-driven evaluations that a middle-ground approach outperforms those two de facto isolated views of learning for network management.

3.4.1 Requirements in D2.2

The mapping between the requirements established for forecasting solutions in D2.2 [1] and the proposed model is presented in the following table.

Table 17. *Requirement satisfaction of optimistic learning-based management.*

Requirement from D2.2 [1]	Rationale
FR-CFORE-000	This NI Functionality addresses one specific problem in the context of capacity forecasting, hence contributes to meeting the requirement of designing models for capacity prediction that support Network Intelligence (NI) algorithms across the mobile network architecture.
FR-CFORE-003	DAEMON's CFORE shall operate over streaming data. Indeed, the proposed solution is amenable to real-time execution, without requiring offline training, and update its operation mode based on the observed streamed datapoints.
FR-CFORE-005	DAEMON's CFORE shall be able to learn autonomously their objective/loss function. The proposed solution contributes in that direction as it optimizes the learning objective based on the forecasting accuracy towards achieving the set performance requirements. This, essentially, amounts to an automatically-selected learning loss objective

3.4.2 Guidelines from D2.2

This solution for this NI functionality abides by the guidelines established in Section 4.1 of D2.2 [1], in terms of adaptation of legacy AI models in order to increase their robustness and adaptation to streaming data. In particular, the optimistic learning techniques proposed in this thread combine deep-learning-type of models with online learning approaches through meta-learning (via regularization or expert-model mixing) in order to achieve fast learning and robust learning at the same time. This departs from previous approaches that rely only on AI-based forecasting for network actions (hence, ignoring readily available data at the runtime) or only on online learning techniques which ignore offline and prior available datasets.

3.4.3 Solution description

In D4.1 [3], Section 4.4, we described the problem of prediction-assisted resource scheduling for intelligent services that exhibit novel and previously unseen challenges in terms of the formulated optimization problems that one needs to solve. We outlined there our goal to develop a novel ML-based optimization framework that allows us to maximize the performance of these services while handling uncertainties in the request pattern and in the (possibly time-varying) objective functions. We explained in detail the importance of these problems by highlighting the gap in the literature on network resource management techniques. In the following, we describe the technical approach we followed to tackle this problem, leveraging and extending the theory of optimistic learning.

Our contributions can be separated into two different thrusts. The first achieves the desired learning objective through proper regularization of the learning functions, while the second thrust achieves the same goal by using the celebrated experts-learning model. Our aim is to learn fast the optimal resource orchestration solution for these scheduling problems whenever the predictions we have at our disposal are accurate while maintaining the worse-case learning rates in the scenario the predictions are inaccurate or even adversarial.

3.4.3.1 Optimistic Learning through Adaptive Regularization

We demonstrate our ideas and results using the paradigm of edge caching, which is an umbrella term for problems related to content caching, caching of services and code, architectures related to edge computing and so on. The system model, in any case, can be abstracted by the following diagram, which shows a set of servicing points (edge caches) that can potentially serve requests from a set of user locations. Each user can be served by multiple caches, possibly with different costs (based on channel conditions, physical distance, etc.), and there is a root server that handles all the unserved requests. In

this scenario, we also have a recommender system, as those that appear naturally in content platforms like YouTube and Netflix, which provides viewing recommendations to the users. These recommendations can naturally serve as predictions for the users' requests who indeed tend, to some (unknown) degree, to follow such provided recommendations. In other cases, the predictions might be provided by a general forecasting model, e.g., an LSTM-based model.

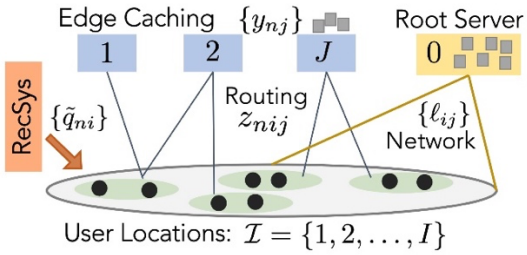

Figure 9. *A Set of user locations generate requests for services or content, which are routed, based on the learning algorithm, to one of the edge servers. The RecSys represents a potential recommender system that can offer suggestions for services or files and, in turn, serves as a predictor as its suggestions are used by the algorithm as predictions for the next-slot requests.*

The goal of the system is to make routing and caching decisions in order to maximize the performance of the service (i.e., maximize the cache hits), without, naturally, having access to the future requests of the users. Ideally, if the system had an oracle that predicts all the future requests perfectly, we could solve the max-hit optimization problem, which is a static optimization problem, to find the optimal cache configuration and routing plan; this is our benchmark which can be written as follows:

$$\max_{x} \sum_{t=1}^{T} f_t(x) \; s.t. \;\; z_{nij} \leq y_n \ell_{ij}, \;\; z \in Z, \; y \in Y$$

where $f_t(x)$ is the objective function that captures whether we had a cache hit or miss based on each slot's user requests. And note that the routing variables z and the caching variables y are subject to the respective cache and link capacity constraints, Z and Y , respectively.

In the absence of an oracle, however, we rely on an online learning algorithm that achieves, asymptotically, as good performance as that benchmark. The figure below summarizes the model for the algorithm's execution. The utility functions capture the desired performance of the system. RecSys is a recommender system which, in the context of modern content delivery systems, can serve as a predictor, i.e., a forecaster (in general, it can be any model proposing services or content to users), and the accuracy of the predictions is accounted for through the regularizer.

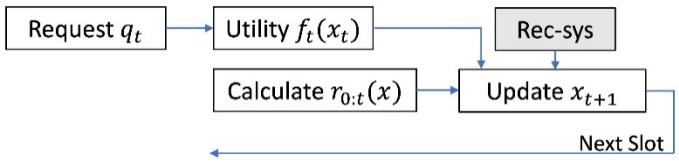

Figure 10. *Key building blocks and operation model for prediction-assisted learning.*

The algorithm execution is summarized in the following template:

Algorithm 1: Optimistic Bipartite Caching (π_{obc})

Input: $\{\ell_{ij}\}_{(i,j)}$; $\{C_j\}_j$; $\mathcal{N}$; $x_1 \in \mathcal{X}$; $\sigma = \sqrt{2}/D_{\mathcal{X}}$.
Output: $x_t = (y_t, z_t), \forall t$.
for $t = 1, 2, \ldots$ **do**
    Route request q_t according to configuration x_t
    Observe system utility $f_t(x_t)$
    Observe the new prediction $\widetilde{c}_{t+1}$
    Update the regularizer $r_{0:t}(x)$ using (7)-(8)
    Calculate the new policy x_{t+1} using (9)
end

Figure 11. *Algorithm template for optimistic-learning based caching in bipartite networks.*

Where the basic iteration for the joint routing and caching policy consists of the following rule:

$$x_{t+1} = \arg\min_{x} \{r_{0:t}(x) - (c_{1:t} + c'_{t+1})^{\mathsf{T}} x\}$$

where c'_{t+1} is the prediction for the gradient vector in the next slot. And our goal is to achieve regret that grows sublinearly, namely:

$$R_T = \sum_{t=1}^{T} f_t(x^*) - \sum_{t=1}^{T} f_t\,(x_t)$$

where x^* is the optimal solution one could have found if there was an oracle that provided all the future cost functions, across all T slots, from the beginning of the time period. Clearly, this solution is not available, but yet we prove that our algorithm can perform as well as this solution. Indeed, our main result is the following theorem:

Theorem 1. Algorithm 1 ensures the following regret bound:

$$R_T \leq 2\sqrt{1 + JC\sum_{t=1}^{T}|c_t - c'_t|^2}$$

This result states that whenever the predictions c' for the next-slot gradient $c = \nabla f_t\,(x_t)$ of the cost functions are perfect, we manage to achieve constant regret. Hence, its average diminishes really fast, while in the worst case, we still achieve sublinear regret, as would have done a learning algorithm without predictions. In other words, we have succeeded incorporating the forecasting model without risking the robustness of our online learning algorithm.

3.4.3.2 *Optimistic Learning through Experts Mixtures*

Next, we change gears and follow a different approach to incorporating predictions in our learning algorithm. The high-level view of our approach is depicted below.

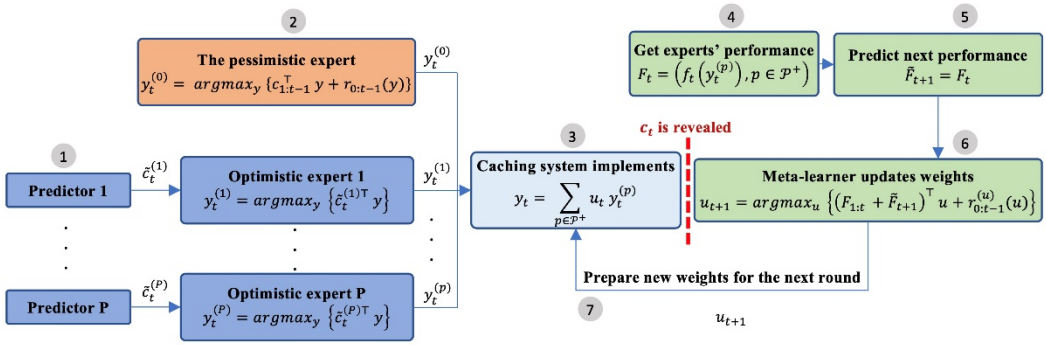

Figure 12. *Experts-based model for incorporating multiple predictors in an online algorithm.*

In this approach, we use the experts' model where a meta-learner takes recommendations from one or more predictors about the optimal caching policy for the next slot. At the same time, one of the experts is a standard robust online learning algorithm, such as Mirror Descent, FTRL, and so on. The meta-learner observes in each round the suggestions of the predictors and gradually adjusts its selection towards those that tend to give more accurate recommendations. This framework provides an easy way to incorporate predictions into the learning model and offers provably the same performance guarantees as the above model while allowing us to use more than one predictor at the same time.

3.4.4 Mapping to major standards (if applicable) and gaps identification

This is a generic NI algorithm, following a different rationale than previous AI or "pure" online learning algorithms, that could be deployed at different network equipment, datacenters, or edge devices. It is thus a module that could be implemented in the context of, e.g., O-RAN.

3.4.5 N-MAPE-K representation

The following table contains the decomposition of the predictions-assisted learning solution into the different components of the N-MAPE-K representation.

Table 18. *N-MAPE-K representation for the optimistic learning-based network management.*

N-MAPE-K module	Solution module
Monitor	The current traffic demands, link capacities, energy consumption, and in general the metrics of interest to be optimized. Also, monitor the forecasting output.
Analyze	Historical measurements of traffic, link capacities, energy consumption, etc. Similarly, need to calculate the forecasting error.

Plan	The NI optimistic-learning algorithm is trained in an online fashion and leverages inputs from a forecasting model (or, an oracle) and determines the next network control decision in the time scale of reference.
Execution	Allocate resources (e.g., compute or network bandwidth) based on the algorithm's output. This can be realized, e.g., in O-RAN, by placing content items of interest at edge servers or code libraries at MEC systems.
Knowledge	The economic OPEX cost of the allocated capacity and the monetary penalty induced by the selected allocation, as computed by the service provider based on QoE/MOS information of the end users and according to the NSSI service-level agreement (SLA).
Training Loss / State – Actions – Rewards	The NI algorithm is a hybrid online/offline model that uses real-time observations and forecasts in order to maximize the performance of the network while being robust to inaccurate forecasting models.

4 Automated anomaly response

In this section, we update our work on the topic of anomaly detection for different communication systems. We present four different solutions that tackle anomalies in different parts of the cellular ecosystem, from the RAN and base stations to the international interconnection network between mobile operators.

4.1 Federated learning powered anomaly detection in B5G/6G

In D4.1 [3], Section 5.1, we described the problem of detecting, in an efficient manner, anomalies in B5G/6G infrastructures that are deployed to serve diverse verticals. These verticals will be requesting a set of services, which can generate diverse traffic profiles. For instance, a utility can generate traffic from sensors, video streams, or even audio; these streams can vary in requirements, for instance, in terms of delay and reliability. Systems should be prepared to handle situations that exhibit a behavior beyond the ordinary or agreed ones. This is important for preserving the services of all verticals served by a network segment. The reason for the unordinary behavior can be due to malevolent reasons or to something else, e.g., some device malfunctioning, extraordinary requirements from the vertical, underestimation (or overestimation) of resources, etc. As such, the problem falls in the class of problems denoted as anomaly detection. In this solution, we present a Federated Learning approach for anomaly detection, which builds upon different ML algorithms and approaches characterized by low complexity and selected so as to optimize the system efficiency in terms of high scalability, data privacy, low network traffic monitoring overhead and low energy costs.

4.1.1 Requirements in D2.2

The mapping between the requirements established for the Federated Learning (FL)-powered Anomaly Detection solution in D2.2 [1] is presented in the following table.

Table 19. *Federated Learning-powered Anomaly Detection. Requirement satisfaction.*

Requirement from D2.2 [1]	Rationale
FR-AARES-000	This NI Functionality uses Federated Learning-powered anomaly detection capabilities to identify diverse types of anomalous network behaviors related to KPIs and computing resource usage.
FR-AARES-001	This NI Functionality adapts the local anomaly detection process frequency, based on the incoming data volumes per FL client. The centralized anomaly detection model is also independently updated on a configurable basis, regardless of the client ratio that has provided its locally trained model's output.
FR-AARES-003	This NI functionality relies upon a Federated Learning-based approach towards high scalability, without the need to increase the network traffic monitoring overhead; additionally, the low complexity ML model that has been employed leads to low energy costs per client, but also at the central FL controller.

4.1.2 Guidelines from D2.2

The design of our solution has followed the guidelines established in D2.2 [1], Section 4.1 and 4.2. Those concern the incorporation of prior knowledge in decision-making schemes: the Federated Learning setting enables each FL client to extract knowledge from its own specific datasets and exchange it with other clients via a centralized learning process, which ingests complementary knowledge from all clients; in addition, the guidelines also refer to the support for decentralized and distributed data management, for heterogeneous devices, and for different time scales: those are enabled by the independent local operation of each one of the clients. Thirdly, the guideline about the low inference time and the energy consumption is addressed via the adoption of low complexity ML models, both in the local (DBSCAN algorithm), as well as the centralized (CART algorithm) learning processes. Last but not least, the Federated Learning-powered NI functionalities guideline is addressed, via several intelligent agents deployed in the various FL setting clients acting cooperatively, via sharing their DBSCAN-based outliers and border points, which are described in the following in detail.

4.1.3 Solution description

One of the most widely investigated approaches in the problem of anomaly detection problems is clustering. Density-based spatial clustering of applications with noise (DBSCAN) is a classical approach that has been applied in several outlier detection problems. DBSCAN performs unsupervised learning and classifies the analyzed data points into core points (normal behavior), border points (potentially anomalous behavior) and outliers (anomalous behavior).

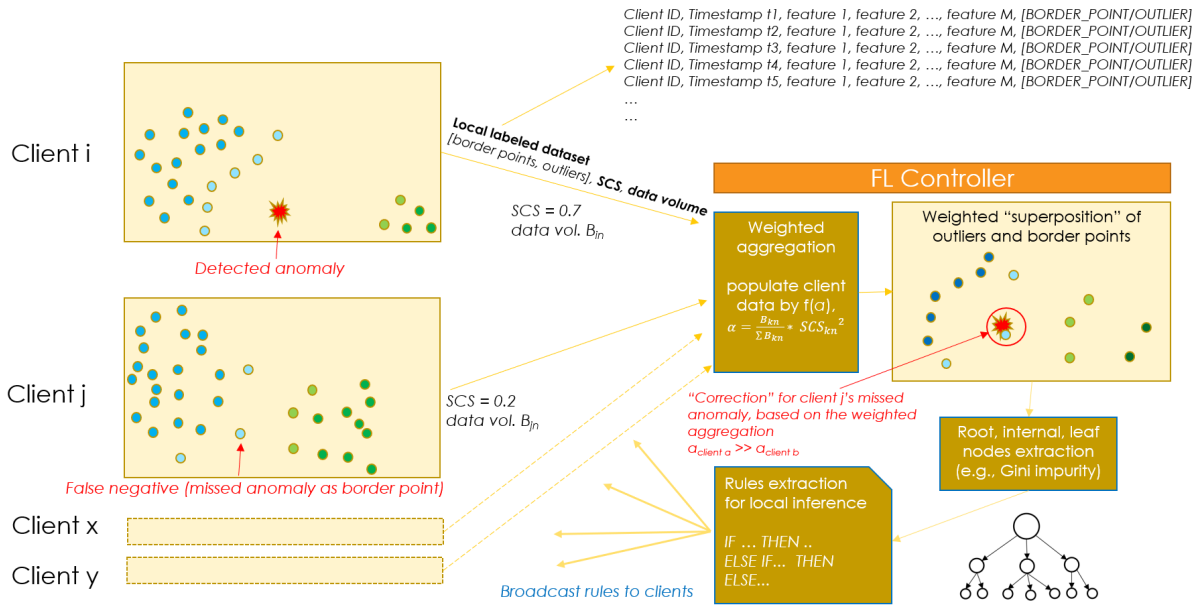

Figure 13. *Overview of the Federated Learning-powered Anomaly Detection solution.*

The first step of the whole process is to configure the DBSCAN parameters, namely the values of ε (i.e., the minimum distance that shall separate the two nearest points in different clusters) and the minPts (i.e., the minimum number of points required to form a dense region). This configuration is dataset-specific and depends on the investigated feature vector size. Then, the DBSCAN is triggered for each FL client to execute locally based on the latest data received. Each client performs a distributed, "own-evaluation" of the process output, via calculating the Silhouette Coefficient Score (SCS). Then, only if the SCS indicates a satisfactory value (i.e., positive value and close to 1), then the respective client is requested to transmit the DBSCAN's output to the central FL controller, i.e., the extracted outliers, border points, SCS value and data volume. The latter is important in the centralized process, as the data volume that has been processed by each one of the FL clients, defines its respective "weight" in the federation process (algorithm pseudocode follows for more information).

Specifically, the centralized step adjusts the population of the received datasets, (i.e., the vector of the outliers, border points, SCS value, and data volume received from each FL client by a factor $\alpha = \frac{B_{kn}}{\sum B_{kn}} * SCS_{kn}{}^2$. Here B_{kn} refers to the logged/processed data volume of client k, during epoch one training period, over the sum of all FL clients' data. Then, the CART algorithm is used on the adjusted data; CART is a *Gini impurity index*-based decision-tree extraction algorithm, which extracts a number of rules that are then broadcasted to the FL clients of the system.

In this hybrid approach, the FL clients are applying rule-based anomaly detection for real-time operation, while the unsupervised DBSCAN is used at different timescales and requests for new incoming data towards enhancing the centralized CART model.

4.1.4 Mapping to major standards (if applicable) and gaps identification

This is a generic NI algorithm that could be deployed at different network equipment types. It is thus a module that could be implemented as a xApp/rApp in O-RAN or as a federated learning architecture approach, as introduced in Rel.17 in the context of NWDAF specifications, by 3GPP.

4.1.5 N-MAPE-K representation

The following table contains the decomposition of the FL-powered Anomaly Detection solution into the different components of the N-MAPE-K.

Table 20. *Federated Learning-powered Anomaly Detection. N-MAPE-K mapping.*

N-MAPE-K module	Solution module
Monitor	Each FL client monitors own compute resources (CPU, memory) and network related KPIs (DL/UL throughput, latency, packet error rate, jitter)
Analyze	The DBSCAN process executed locally extracts normal, suspicious, and anomalous behavior, defined in the local context.

Plan	The CART algorithm applied centrally processes the populated, weighted [outlier, border point, SCS, data volume] information, as aggregated from all clients and extracts the rules to be forwarded to the clients.
Execution	Each FL client executes supervised anomaly detection locally based on the received CART model from the FL controller.
Knowledge	Compute resource, network related KPIs, data volume per client, DBSCAN performance/dataset homogeneity per client
Training Loss / State – Actions – Rewards	The centralized FL controller is responsible for extracting the general anomaly detection rules to be applied as actions by the FL clients towards local supervised anomaly detection. Data volume and quality of the local DBSCAN operation results in higher weight/reward for the specific client.

4.2 Proactive anomaly detection for IoT services in a global roaming platform

ANCHOR integrates a set of advanced deep learning solutions for anomaly detection for IoT devices. We designed the solutions using only control plane information (i.e., signaling information). We then explore two major ways of encoding the data, namely volume-based matrices (image-like encoding) or sequences of protocol messages. Given the heterogeneity of the IoT devices that the IoT platform connects, we focus on the data pre-processing, feature engineering, and unsupervised clustering of devices. Our goal is to build a generalizable solution that applies to any IoT platform, regardless of the IoT verticals it serves. We will present the validation results for our approaches in D5.2.

4.2.1 Requirements in D2.2

We present in the following table the mapping between the requirements we identified in D2.2 [1] for Automated Anomaly Response (AARES) and the solutions we designed for *anomaly detection for IoT services in a global roaming platform (ANCHOR)*.

Table 21. *Anomaly detection for IoT services in a global roaming platform (ANCHOR). Requirements satisfaction.*

Requirement from D2.2 [1]	Rationale
FR-AARES-000	The ANCHOR NI functionality integrates a set of anomaly detection approaches to identify different types of device-level anomalies for IoT verticals that depend on Telefonica's global roaming platform to manage the connectivity of their devices.
FR-AARES-001	ANCHOR adapts the anomaly inference frequency depending on the needs of the engineers within the Global Roaming Technical Unit (GRTU).
FR-AARES-002	ANCHOR relies on a vast dataset that we collect from the operational roaming platform for IoT that the GRTU within Telefonica Global Solutions operates. For the ground truth dataset collection, we rely on the ticketing system of the operational team who manages the system.
FR-AARES-003	ANCHOR is designed to comply with the operational restrictions from the GRTU. This allows them to integrate the ANCHOR software in the cloud-based big data platform that they use internally.
FR-AARES-004	We enable the GRTU to retrain the ANCHOR models periodically (e.g., monthly), to accommodate for changes in the feature distributions.

4.2.2 Guidelines from D2.2

ANCHOR integrates tailor-made NI solutions for anomaly detection in the connectivity status of cellular devices that map to different IoT verticals (e.g., smart elevators, fleet tracking solutions, cargo ships). We designed the solutions through periodic interactions with the GRTU team within Telefonica Global Solutions, which operate the IoT-managed connectivity service for the unit. In this section, we augment the guidelines (Table 13 in D2.2 [1]).

Based on our work, we have found that applying ML solutions out of the box on curated datasets streams that we collect directly from the system is not an optimal approach. Our experience shows that, in the case of anomaly detection, in particular, data quality carries an elevated significance due to the NI's heightened downstream impact, which translates to an impact on the high-value mobile operators' interconnection agreements, IoT application requirements of security and privacy. Thus, AI models for AARES shall rely on high-quality tailored datasets with features based on expert domain knowledge.

Moreover, our effort with ANCHOR highlights that AARES NI on integrating the idea of data excellence as a first-class citizen of AI. This is especially important when attempting to integrate within communication platforms advanced AI models initially designed for other specific areas (e.g., image processing). We found that data quality issues in AI easily cascade and impact the quality of the end results. Thus, our research underscores the need for data excellence in building NI solutions for AARES and a shift to proactively consider diligence in data as a valuable contribution to the AI ecosystem. Specifically, when dealing with sensitive real-world systems (e.g., an international platform that connects IoT devices corresponding to multiple types of IoT applications), one must look for the least intrusive dataset that can allow a meaningful representation of the state of the system.

Finally, when testing some of the ANCHOR solutions with the GRTU's engineers, we have identified an important challenge regarding the trust between the human expert user and the NI functionalities that we should address.

4.2.3 Solution description

We evolved the ANCHOR solution from the architecture we presented in D4.1 [3] (Section 5.2) in two ways: (i) we improved the clustering approach for devices by relying on multiple features we derived to capture the behavior of IoT devices (and not just the sheer volume of signaling traffic; (ii) we integrate within ANCHOR a new NI approach for anomaly detection that relies on seq-2-seq models.

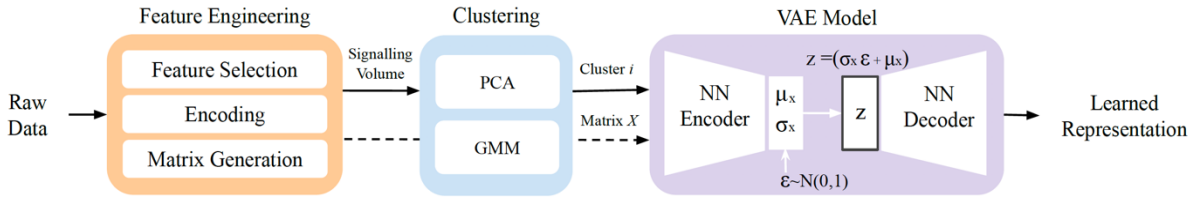

***Figure 14**. The proposed anomaly detection framework for ANCHOR as per D4.1 [3].*

Updated Clustering Approach. In the original ANCHOR solution approach (see Figure 14), we relied on clustering based only on signaling volume to avoid confusing the model (e.g., the signaling traffic volume of a connected car is different than that of a connected smart meter, so we must separate them to ensure that the unsupervised model learns the correct baseline behavior). However, we found that separating devices based on the sheer volume of signaling traffic is insufficient to ensure uniformity in the clusters of devices we monitor. For example, in Figure 15 we show that IoT customer B also exhibits weekly patterns within their Diameter signaling traffic compared to IoT customer A, which does not. This hints that the periodicity of the data connections must be considered when clustering the IoT devices.

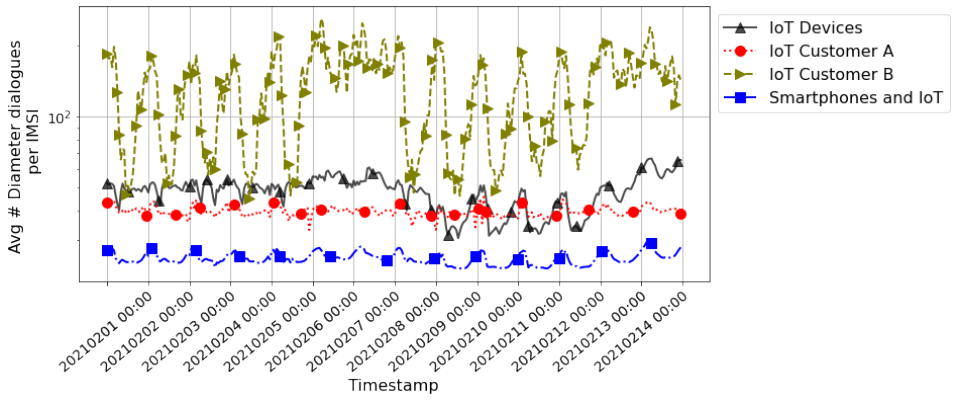

***Figure 15.** Difference in traffic patterns of distinct IoT customers mapping to different IoT applications.*

We thus update our clustering approach to integrate a wider set of features, which we will further report in detail in D5.2.

ANCHOR updated framework. Our initial evaluation results (as presented in D5.1 [21]) showed that the CNN-VAE model performs best in anomaly detection for IoT devices corresponding to one specific client. However, we have observed that many of the anomalies we captured in our ground truth dataset did not come with a very high anomaly score.

Based on our understanding and in-depth analysis of the signaling behavior of the IoT devices, we believe that it is important to also teach the anomaly detection models the sequence of the signaling messages and their frequency, apart from the signaling volume. Thus, as part of ANCHOR, we integrate a new Sequence-to-sequence (seq2seq) model that can understand and learn the sequence of the signaling messages within the traffic we monitor.

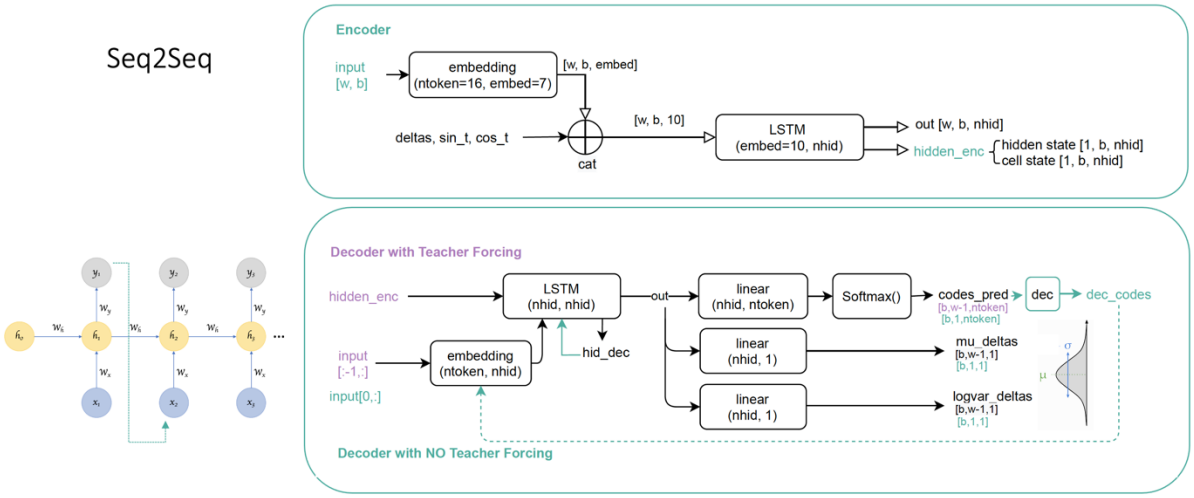

Figure 16. *Seq2Seq model for ANCHOR; this model is an alternative approach to the CNN-VAE model we previously included in the ANCHOR framework.*

The data encoding for the seq2seq model is different than the traffic volume matrices that we were using in the case of the CNN-VAE model. Specifically, we use as input for the seq2seq model the stream of signaling data, considering the signaling request code, its result code, the duration of the signaling dialogue and the time delta from the previously seen signaling dialogue from the same device. This allows us to capture the logic in the sequence of signaling requests and the frequency of the signaling dialogues.

We are now in the process of evaluating the performance of this new model, and we will provide the evaluation results in D5.2.

4.2.4 N-MAPE-K representation

Table 22. *Anomaly detection for IoT services in a global roaming platform (ANCHOR).*

N-MAPE-K module	Solution module
Monitor	The IoT device-level signaling traffic (D8 in D5.1 [21]) from the signaling points integrated within the IoT platform service of the roaming platform. This corresponds to the dataset we reported as D8 in D5.1, which we collect in almost real-time (every 5min) from the operational roaming platform.
Analyze	We create a large set of features based on the signaling dataset (D8 in D5.1 [21]) that capture the connectivity status of the IoT device. With this, we generate clusters of devices with similar historical behavior (regardless of the IoT application). We then encode the signaling data into matrices (image-like data encoding) to further run the anomaly detection.
Plan	At the time of writing, ANCHOR integrates two NI solutions: a CNN-VAE approach and a seq-2-seq approach. Both approaches rely on observing the activity of devices (within the same cluster of behavior) during a period of one month to learn their baseline behavior.
Execution	Announce anomalies based on divergence from the baseline learned distribution learned from historical data. Specifically, we use the KL divergence, which corresponds to the difference between the learned latent space and the normal distribution (i.e., being zero if the compared distributions are identical and greater than zero depending on how much they differ). The higher the KL divergence score, the more probable the anomaly.
Knowledge	The ticketing information from the GRTU corresponding to anomalous events that affect the system to various extents. A high-quality ground truth dataset that continuously updates to integrate a wide variety of anomalies ensures the detection of similar anomalies.
Training Loss / State – Actions – Rewards	The ANCHOR NI solutions are pre-trained unsupervised models validated based on data stored in the knowledge.

4.3 Anomaly forecasting in mobile service traffic at base station level

In this use case, we consider an anticipatory anomaly detection problem, where the NI must trigger an alarm when an abnormal future traffic load is expected for a specific Network Slice Subnet Instance (NSSI). The anomaly detection problem is summarized in Figure 17(a). A predictor module is in charge of producing a probability distribution of the traffic demand that the target service will generate in the next time slot. Such a probabilistic prediction is compared against a reference interval that encompasses the expected range of normal traffic values in the following time slot. Then, if the probability of the anticipated traffic being outside the reference interval is beyond a threshold, an alarm is raised. This allows the associated NI to perform some preventive actions, such as those detailed in the 3GPP TS 23.288 technical specification under the "Abnormal UE behavior" analytics, which captures anomalies such as unexpected large rate flows generated by terminals.

4.3.1 Requirements in D2.2

The mapping between the requirements established for capacity forecasting solutions in D2.2 [1] and the proposed model is presented in the following table.

Table 23. *Anomaly forecasting. Requirement satisfaction.*

Requirement from D2.2 [1]	Rationale
FR-AARES-000	This NI Functionality addresses a subset of the tasks involved in the broader requirement of automatically detecting, analyzing, and acting against network anomalies. Specifically, it targets the automated detection and prediction of such anomalous events.
FR-AARES-002	This NI functionality leverages a statistical approach to identify anomalies, which combines with probabilistic predictions. For model training, the solution thus has specific data requirements, including a sizable amount of historical data to establish normal behavior and ground truth occurrences of anomalies (in terms of the reference parameter *R*).
FR-CFORE-004	This NI functionality leverages an efficient Bayesian approximation technique via dropout layer activation during inference to produce a probabilistic output for a prediction task. It thus provides a way to implement this capacity forecasting requirement.

4.3.2 Guidelines from D2.2

The NI solution proposed for this use case follows the guidelines established in Section 4.2 of D2.2 [1], by investigating alternative methods that allow for broadening the spectrum of learning and optimization tools for practical NI tasks. Indeed, the model employed to perform the anticipatory anomaly detection task that is part of the use case adheres to the full hybrid approach that is proposed in that Section 4.2.4 of D2.2 [1], which is a better alternative to pure deep learning AI models commonly used in the literature for network traffic forecasting.

4.3.3 Solution description

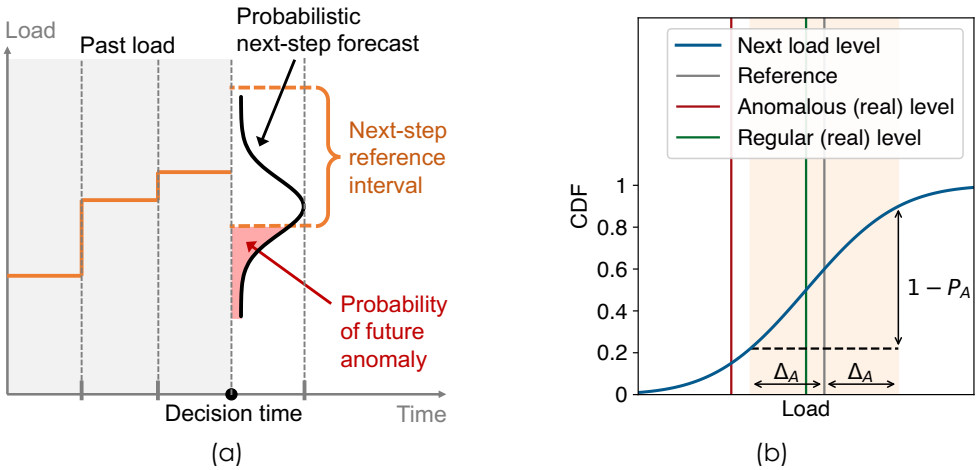

Figure 17. *Anomaly forecasting use case: (a) Representation of the anomaly forecasting problem; (b) Explanation of the operation of anomaly detection.*

To address this problem, we propose a simple yet practical implementation of the approach above that is commonly adopted in many fields also outside networking [22]. First, it is worth noting that the output of the forecasting algorithm shall not be a scalar but a probability distribution of the future traffic load. This type of output is implicit in certain types of models like Bayesian Neural Networks, which are however computationally expensive and not suited for resource-constrained network environments. In order to generate a probabilistic forecast with a generic neural network, we resort to recent findings in uncertainty modeling [23]: specifically, by activating dropout layers in the predictor during the inference phase and performing a Monte Carlo testing, the neural network returns a set of values that has been shown to closely approximate the probabilistic result of a deep Gaussian process implemented with a Bayesian network.

The anomaly detection algorithm then operates on the empirical PDF f_P of the predicted traffic values for each decision interval, as illustrated in Figure 17 (b). First, the upper and lower limits that mark the boundaries between regular and anomalous values are computed as $x_{l,h} = R \pm \Delta A$, where R is a configurable reference value. From these two values, the probability of a future anomaly is empirically calculated as $P_A = 1 - (F_P(x_h) - F_P(x_l))$, where $F_P(x) = \Sigma_{k<x} f_P(k)$ is the CDF of the anticipated traffic. Finally, an alarm is triggered if $P_A > \tau_A$. The parameters R, ΔA and τ_A control the sensitivity of the algorithm.

The correctness of the anticipatory anomaly detection can be determined by checking whether the actual traffic falls into the $x_{l,h}$ interval or not and computing precision and recall scores. Clearly, a higher accuracy in the probabilistic traffic forecast, denoted by a lower variance around a value closer to the true one, yields better performance: a Mean Square Error (MSE) loss function is thus a sensible choice for model training in this use case.

For the implementation of the predictor, we resort to the DAEMON-proposed TES-RNN model [19] a time series forecasting model which combines a classical Exponential Smoothing (ES) statistical model with a Recurrent Neural Network (RNN) architecture. A complete description of the TES-RNN model is provided in Section 4.2.4 of D2.2 [1].

4.3.4 Mapping to major standards (if applicable) and gaps identification

This is a generic NI algorithm that could be deployed at different network equipment or datacenters. It is thus a module that could be implemented as a xApp/rApp in O-RAN or NWDAF in 3GPP. Therefore, the solution closes an algorithmic gap in terms of NI, and not a standardization gap.

4.3.5 N-MAPE-K representation

The following table contains the decomposition of the anticipatory anomaly detection solution into the different components of the N-MAPE-K representation.

Table 24. *Anomaly forecasting. N-MAPE-K mapping.*

N-MAPE-K module	Solution module
Monitor	The current traffic demands, e.g., measured in Mbps, generated by the target NSSI at the datacenter where the anomaly detection must be performed.
Analyze	Current and historical NSSI traffic demands are concatenated to produce a time series of demand values at the target network datacenter. The reference and sensitivity parameters of the model are derived from historical data.
Plan	The NI algorithm is a TES-RNN model trained with an MSE loss function, which is combined with active dropout layers during inference and outputs a probabilistic prediction of future traffic demands. A statistical logic then decides whether an anomaly will occur in the next time step and generates the final output of the plan phase.
Execution	Announce anomalies based on the forecast and statistical decision-making.
Knowledge	The actual traffic demands, e.g., measured in Mbps, generated by the target NSSI at the datacenter where the anomaly detection must be performed must be stored and used as the ground truth for training.
Training Loss / State – Actions – Rewards	The NI algorithm is a pre-trained supervised model using an MSE loss over labeled data stored in the knowledge.

4.4 Noisy Neighbours in vRANs

The maturity of Software-Defined Networking and Network Function Virtualization (SDN/NFV) in other domains has spurred the market to build virtual network functions (VNFs), such as firewalls, switches, VPNs, etc., that provide carrier-grade performance.

However, research has shown that resource contention between VNFs sharing computing infrastructure may lead to up to 40% performance degradation compared to dedicated platforms, even when using dedicated cores. The term noisy neighbor problem has been coined to refer to this issue. In detail, the noisy neighbor's effect arises as the result of non-isolated resources being overused by different tenants so that the usage of those resources restricts tenants' resources by another tenant in such a way as to impact performance negatively.

The virtualization of base stations (vBSs) is not alien to this issue. We confirm this with our findings from experiments in a proof-of-concept vRAN system comprised of instances of a full-fledged 3GPP rel.10 compliant vBS implemented by srsRAN. We deploy each vBS instance into a Docker container allocated to a pool of dedicated CPU cores from an Intel core i7-7700K CPU @ 4.20GHz in a shared off-the-shelf server. We then initiate bidirectional data flows, both uplink (UL) and downlink (DL), with maximum load and good wireless channel conditions between each vBS instance and a (different) legacy user equipment (UE).

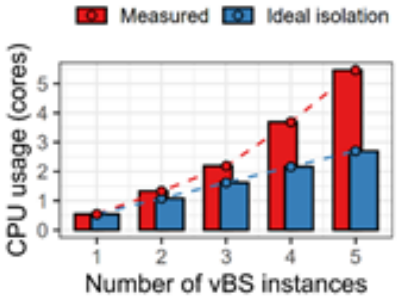

***Figure 18**. vRAN per-core CPU usage with # of vBS.*

Figure 18 depicts the CPU usage of the whole vRAN as a function of the number of vBS instances deployed. The bars in blue depict the expected CPU consumption provided with perfect resource isolation. We compute these by linearly scaling up the CPU usage of a vRAN by one vBS instance. The red bars show the actual measurements, which unveil an additional CPU overhead induced by the aforementioned resource contention in imperfectly isolated computing platforms, and moreover, this overhead increases even exponentially with the increasing number of vBS instances.

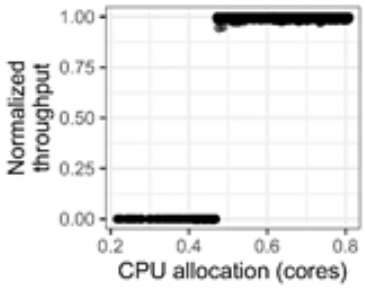

***Figure 19**. Throughput vs. CPU allocation.*

In this context, exploring the gain of network function virtualization in the RAN arena may prove challenging. First, the real-time nature of the workload spawned by vBSs makes them more sensitive to noisy neighbors. Figure 19 shows the attained throughput performance (normalized) for different CPU allocations (x-axis). In line with our earlier results, throughput rapidly collapses as soon as there is a deficit – even if minimal – of computing resources. This occurs because the physical layer (PHY) of vBSs has tight frame processing constraints. When violated, users lose synchronization with the vBS, resulting in connectivity loss. This provides evidence that, while traditional VNFs get a performance degradation that is somewhat proportional to the deficit of computing resources, vBSs are inelastic and hence, substantially more sensitive to resource contention.

An in-depth experimental analysis will be provided in deliverable D5.2.

4.4.1 Problem description

Although the noisy neighbor's effect is particularly difficult to quantify, it is crucial to take it into account when optimizing a vRAN system as ignoring so may lead to disastrous consequences. It is key to develop an effective solution to adapt the computing capacity to the requirements of the different vBSs deployed so it is possible to serve the traffic demands. A shortage of computing resources (due to the influence of noisy neighbors or something else) could cause that the users associated with vBSs in the system lose synchronization, induce a high number of radio link errors, cause an increased end-to-end latency with high jitter, and, ultimately, produce a data plane reset.

To begin with, the amount of CPU resources required by a single vBS instance depends on the network traffic demand on both DL and UL directions, the signal-to-noise ratio (SNR) of each wireless link and the associated Modulation Coding Scheme (MCS) used for communication, in a non-trivial manner. Moreover, estimating the actual requirements for a set of vBS instances sharing a platform is even more challenging because the overhead introduced by computing resource contention (noisy neighbors) depends on the computing cores used to process each vBS workload, the amount of isolation across vBS instances, and the maximum computing capacity available.

On the one hand, over-dimensioning the allocation of computing resources incurs high infrastructure costs as many computing cores might not be needed when running a small number of vBS instances or when the aggregated load is low, and the electricity bill associated with unneeded active cores can be substantial. On the other hand, pooling a reduced number of cores across many instances (i.e., forcing vBSs to share) may lead to throughput loss because heavy resource contention leads to severe computing overheads. Hence, to maximize network performance with minimum infrastructure cost, it is important to tailor the amount of resource isolation appropriately.

We consider an O-RAN cloud computing platform (O-Cloud) providing computing resources for multiple vBS instances deployed therein, i.e., each vBS instance shares the same pool of computing resources. In such a scenario, we aim to solve the problem by devising which computing cores need to be active in the pool to serve the demand of each vBS, which process uplink and downlink traffic, to maximize performance and minimize energy consumption.

The design of our solution will be presented in D3.3, and an experimental evaluation of the solution will be presented in D5.3.

5 Self-learning MANO

In contrast to 4G, 5G and 6G enable the sharing of the network infrastructure among different tenants based on virtualization techniques (Network Function Virtualization (NFV)). Each of these tenants can set up its own network slice by spinning up the required VNFs and routing control and data traffic through them. A consequence of this additional flexibility is that network management and orchestration (MANO) in 5G and 6G become orders of magnitude more difficult and virtually impossible for a human operator to manually perform, hence requiring automated solutions that are flexible enough to adapt to diverse working conditions without human intervention. ETSI has defined several MANO operations that are critical for the correct operation of NFV-based networks [4], amongst them placement and scaling.

5.1 Placement and routing new requests

The method described below is an extension to ETSI MANO [24]. When a new network service needs to be set up, the associated VNFs need to be placed on Commercial Of The Shelf (COTS) servers sitting in datacenters and the control and data traffic flows between these VNFs need to be routed over the network.

In D4.1 [3], a Reinforcement Learning (RL) solution was provided to place the VNFs in one datacenter and route the associated traffic on the intra-datacenter datacenter network. In the present deliverable, we provide a solution to place the VNFs in any of a number of geographically distributed datacenters (also referred to as Points-of Presence (PoPs)) and route the traffic over the inter-datacenter network. In fact, we propose and compare the performance of a centralized single RL agent solution with a distributed multi-agent RL solution. We pay special attention to the kinds and amount of information that needs to be shared between the central controller and between the PoPs in the single-agent and multi-agent solutions, respectively.

5.1.1 Requirements in D2.2

Table 25 shows how some of the requirements related to the self-learning MANO (enumerated in D2.2 [1]) are satisfied by the placement algorithm described in this section.

Table 25. *Requirement satisfaction for SLMANO.*

Requirement from D2.2 [1]	Rationale
FR-SLMANO-000	This placement algorithm is part of autonomous and self-learning orchestrators which can operate with minimal human intervention. It places VNFs associated with network services (e.g., network slices) autonomously.
FR-SLMANO-002	This placement algorithm optimizes an accumulated discounted reward that depends on high-level application properties (i.e., end-to-end delay).
FR-SLMANO-003	The learning is such that it converges to a (reasonable) stable working point.
FR-SLMANO-004	The MARL agents express their wishes (on how much they want to host a VNF) in terms of human-understandable actions. These actions can be explained given the status of the load on the PoPs governed by the agents.

5.1.2 Guidelines from D2.2

The algorithm described next fits in the framework described in Section 4.2.3 "AI-enhanced MANO (A8)" of D2.2 [1]. While in D4.1 [3] a placement (and routing) algorithm for a single datacenter (governed by one Virtual Infrastructure Managers (VIM)) was described, here we describe a Deep Multi-Agent Reinforcement Learning (DMARL) framework that takes placement decisions in an environment where workloads have the possibility to be placed on datacenters that are managed by different VIMs. Particular emphasis is placed on the status information that needs to be exchanged (e.g., via the OSM framework) for MARL agents to make a sensible decision.

5.1.3 Solution description

Network Function Virtualization (NFV) promotes the use of virtual network functions (VNFs), which can be grouped to form a service function chain (SFC). A critical challenge within NFV is SFC embedding (SFC-E), which is mathematically expressed as a graph-in-graph embedding problem. Given its NP-hardness, SFC-E is commonly solved by approximation methods. Yet, the relevant literature exhibits a gradual shift towards data-driven SFC-E frameworks, namely deep reinforcement learning.

In this section, we initially identify crucial limitations of existing SFC-E approaches. In particular, we identify that most of them have their grounds, not on the algorithmic framework per se, but rather on its exercise in a centralized fashion. Thereby, we devise a cooperative DMARL scheme for decentralized SFC-E,

which fosters the efficient communication of neighboring agents. As we will show in the future deliverable D5.2 our simulation-based experimentation (i) demonstrates that DMARL outperforms a state-of-the-art double deep Q-learning algorithm, (ii) unfolds the fundamental behaviors learned by the team of agents, (iii) highlights the importance of information exchange between agents, and (iv) showcases the implications stemming from various network topologies on the DMARL efficiency.

5.1.3.1 Introduction

Increasing processing demands, typically coupled with highly stringent performance requirements, put modern networks under severe stress. To mitigate the risk of under-delivery, domain experts promptly re-conceptualized several architectural components of networks and pushed innovations that render them more agile, cost-effective, and high performing. One such technology is network function virtualization (NFV), which comprises a paradigm shift from hardware-based to VNFs. As such, common network functions can be implemented in the form of virtual machines or containers and deployed on-demand within COTS servers. In addition, multiple VNFs are arranged into an ordered sequence to form an SFC, which in effect enforces an end-to-end flow processing policy.

Nonetheless, instilling such a high degree of flexibility into networks via NFV comes with a certain cost, which in this case is embodied by MANO complexity. Indeed, instantiating, configuring, monitoring, and scaling SFCs comprise only a subset of operations that need to be applied upon these new network constructs. To support such features, the whole NFV endeavor is backed by the NFV MANO framework, which consists in a hierarchical architecture that positions an NFV orchestration (NFVO) module on top of VIMs. Effectively, each VIM operates on a set of COTS servers (which are typically grouped in a datacenter) and has a detailed view of critical local parameters, e.g., resource consumption levels. Yet, useful information from multiple VIMs shall be conveyed to the NFVO layer, since certain actions of the latter might require the coordination of more than a single VIM. For instance, an SFC might consist of VNFs with various location constraints, raising the need for SFC partitioning and placement across many VIMs.

From an algorithmic point of view, SFC-E is proven to be an NP-hard optimization problem [25], which practically implies scalability limitations. As is typical in problems of such computational complexity, many studies model SFC-E as a Mixed-Integer Linear Program (MILP), and subsequently propose heuristics or approximation algorithms that can cope with the scalability issue at the expense of solution quality, e.g., [26] [27]. However, distance from optimality is not their only limitation. More specifically, these algorithms are usually tailored to specific environments, e.g., certain network topologies), constraints and objectives, thereby requiring drastic redesign in case some problem parameters or goals alter. Additionally, they generally assume perfect knowledge with regard to the resource utilization of the physical network, which is highly unrealistic due to high network dynamics. Yet, modeling SFC-E as, e.g., a MILP, is virtually impossible without this assumption. To overcome these limitations, data-driven resource allocation techniques, primarily based on RL, have recently started gaining traction within the problem space of SFC-E, e.g., [28] [29] [30] [31] [32] [33] [34] [35] [36]. This algorithmic shift has grounds for several arguments. First, rapid advancements in the RL field and, in particular, the application of deep neural networks within the RL framework [37], empower it to handle vast state spaces, e.g., by approximating Q-values. Prior to this development, RL methods, such as Q-learning, were unable to cope with large environments due to memory limitations and the inability to learn all entries in the Q-table. Second, RL methods are inherently purposed to design their own optimization policies. They achieve this merely by interacting with the environment upon which they are applied and by receiving a problem-specific reward signal that expresses the decision quality. Given that these features are aligned with the endeavor set out by operators toward zero-touch automation of their networks, it is only natural that RL algorithms have become state-of-the-art SFC-E methods.

It still remains open questions: whether or not the full potential of RL is fully harnessed within SFC-E, and whether or not it is applied in the proper fashion. The relevant literature exhibits certain limitations, the most crucial of which are listed and analyzed below:

- ***Association of intents.*** A critical limitation of centralized SFC-E is the association of the resource allocation intents of the NFVO with the respective intents of the underlying VIMs. That is, as is common in hierarchical resource management systems with global and local controllers, e.g., NFVOs and VIMs, VIMs and hypervisors, k8s master and worker nodes, the global controller queries the local controllers about their available resources and uses this information to compute a resource allocation decision (which is eventually realized by the latter). Effectively, this limits the inherent capacity of the local controllers to express their own resource allocation intents. In fact, global and local intents are not necessarily conflicting; on the contrary, fostering the acknowledgement of local intents can improve the resource allocation efficiency of the overall NFV MANO system, primarily because each local controller has a more precise view of its actual state.
- ***Decision-making with incomplete information.*** The common assumption that a single learning agent, positioned at the NFVO layer, has a precise view over the entire topology is somewhat unrealistic. In fact, the higher we move along the NFV MANO hierarchy, the more coarse-grained

the information we have at our disposal (because of data aggregation), which implies higher uncertainty. Conversely, if a centralized approach admits partial observability, it follows that the decision-making process relies on incomplete information.

- ***Reduced/fixed action space***. Some studies, e.g., [31] [35], employ techniques that force the action space to shrink by, e.g., clustering a set of points-of-presence (PoPs) into a single group or placing the entire SFC within a single PoP. This indicates that the respective RL methods would struggle to obtain efficient SFC-E policies if the action space was large enough to express more embedding alternatives. On top of that, in most cases, the action space is tightly related to topology nodes, e.g., one action per PoP. Hence, the respective methods would perform poorly in the event of dynamic insertion of topology nodes, as these will be treated as unexplored actions.

To address some of these open questions we

- Devise a DMARL framework for decentralized SFC-E. Our DMARL algorithm consists of independent Double Deep Q-Learning (DDQL) agents and fosters the exchange of concise, yet critical, information among them.
- Develop a single-agent DDQL algorithm for centralized SFC-E and compare its performance with DMARL.
- Quantify fundamental rules identified by the team of agents over the course of training.
- Examine the impact of (imperfect) cross-agent communication across various substrate network topologies.

5.1.3.2 Problem formulation

As SFC is an ordered sequence of VNFs, which enforces a flow processing policy, each VNF of the SFC requires computing resources (e.g., CPU) for packet processing. Additionally, connections among consecutive VNFs, henceforth termed as virtual links (vlinks), require network resources (e.g., bandwidth) in order to forward traffic from one VNF to another. Such computing and network resources are offered by multiple datacenters. Specifically, computing resources are offered by servers, while network resources are allocated from physical links. It is also quite common to assume that computing resources are offered by a PoP (e.g., a datacenter). In this case, PoPs typically expose descriptive values about the resources of their underlying servers, such as their average residual CPU capacity.

SFC-E boils down to the problem of computing a mapping between the virtual elements of an SFC and their physical counterparts, while adhering to resource capacity constraints. Specifically, VNFs are assigned to PoPs in a one-to-one fashion, while virtual links are assigned to physical links in a one-to-many fashion. An illustration of an SFC-E problem instance is given in Figure 20. An SFC (consisting of three VNFs) at the top of the figure is considered to be embedded in a six-PoP topology at the bottom of the same figure. Both VNFs and virtual links require resources (dark circular sectors), while PoPs and physical links are characterized by available and allocated resources (light and dark circular sectors, respectively). Further, each SFC contains a source and a destination, while each PoP is represented by its coordinates in a two-dimensional space.

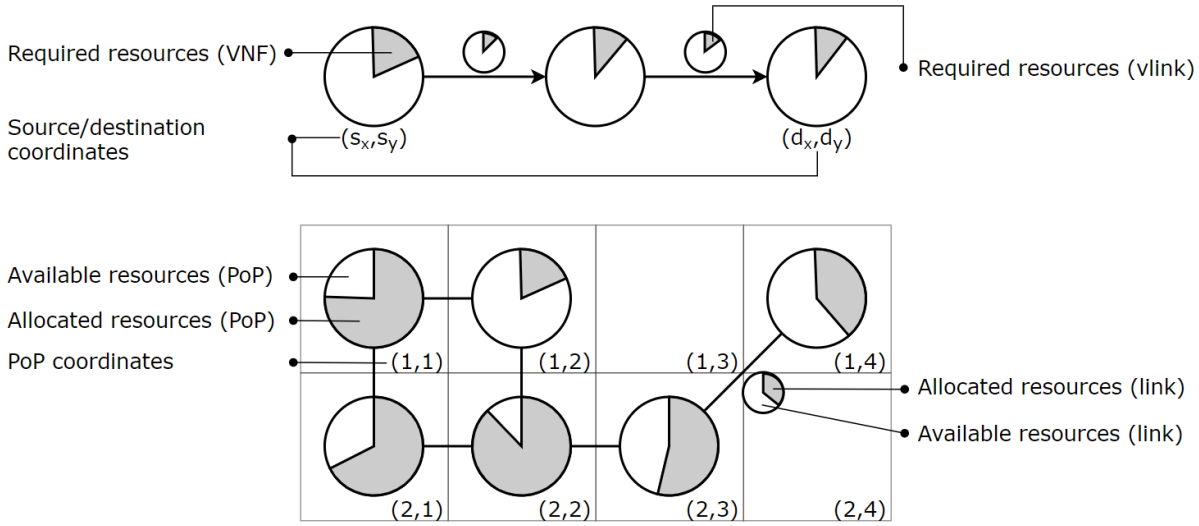

Figure 20. *An SFC-E problem instance. On top, an SFC-R. At the bottom, a multi-PoP topology.*

Our work focuses on the investigation of an RL scheme that is aligned with current system architectures and goals: we opt for the latency minimization objective since a key problem for SFC-E is to fulfill the stringent low-latency requirements.

The system model consists of:

- ***A service function chain request (SFC-R) model***. We model an SFC-R with the 5-tuple $\langle G_R, \text{src}, \text{dst}, t, \Delta t \rangle$. $G_R = (V_R, E_R)$ is a directed graph, where V_R and E_R denote the set of nodes and edges (i.e., VNFs and virtual links) respectively. Additionally, $d_R(i)$ expresses the CPU demand of VNF $i \in V_R$, while $d_R(i,j)$ represents the bandwidth requirements of $(i,j) \in E_R$ (both expressed as percentages). src and dst indicate the source and the destination coordinates of the SFC, while t expresses its arrival time, and Δt denotes its lifespan. We note that both src and dst are modeled as auxiliary VNF nodes with zero resource requirements (i.e., src and $dst \in V_R$ and $d_R(src) = d_R(dst) = 0$). Naturally, this introduces two additional edges to the SFC graph, one from the src towards the first VNF, and one from the last VNF towards the dst (see top of Figure 20. Further, we assume that the src and dst of an SFC correspond to coordinates of arbitrary PoPs, denoted by u_{src} and $u_{dst} \in V_R$.
- ***A substrate network model***. We model a substrate network with an undirected graph $G_S = (V_S, E_S)$ where V_S and E_S denote the sets of nodes and edges. Since we consider G_S to be a PoP-level topology (i.e., a network of datacenters), V_S corresponds to PoPs, and E_S to inter-PoP links. $d_{S,t}(u)$ expresses the available CPU of node u at time t (i.e., the average available CPU of servers belonging to PoP u), and $d_{S,t}(u,v)$ denotes the available bandwidth of the edge (u,v) at time t (again, as percentages). Last, $D(u,v)$ represents the propagation delay of a physical link (u,v).

5.1.3.3 Exact SFC-E

The exact SFC-E is obtained as the solution to the following problem. Let $x_{i,u} \in \{0,1\}$ be a binary decision variable that indicates the assignment of VNF i on PoP u. Similarly, let $f_{(i,j),(u,v)} \in \{0,1\}$ be a binary decision variable expressing the mapping of the virtual link (i,j) on the physical link (u,v). According to previous discussions, the feasibility constraints are the following:

$$x_{src,u_{src}} = x_{dst,u_{dst}} = 1$$

$$\sum_{u \in V_S} x_{i,u} = 1, \quad \forall i \in V_R$$

$$\sum_{u \in V_R} x_{i,u} d_R(i) \leq d_{S,t}(u), \quad \forall u \in V_S$$

$$\sum_{\substack{v \in V_S \\ v \neq u}} \left(f_{(i,j),(u,v)} - f_{(i,j),(v,u)} \right) = x_{i,u} - x_{j,u}, \quad \forall (i,j) \in E_R, \forall u \in V_S$$

$$\sum_{(i,j) \in E_R} f_{(i,j),(u,v)} d_R(i,j) = d_{S,t}(u,v), \quad \forall (u,v) \in E_S$$

$$\sum_{\substack{v \in V_S \\ v \neq u}} f_{(i,j),(u,v)} = 1 - x_{j,u}, \quad \forall (i,j) \in E_R, \forall u \in V_S$$

$$x, f \in \{0,1\}$$

The first constraints enforce the placement of the src and dst VNFs into the respective PoPs. The second constraints impose the mapping of each VNF at exactly one PoP, while the third constraints ensure that PoP CPU is not over-allocated. The fourth constraint associates x and f variables on the grounds of flow conservation. Finally, the fifth constraints dictate that link bandwidth capacities are not violated, the sixth constraints avoid loops in link mappings, and the seventh constraint enforces variable domains.

The objective is to:

$$\text{Minimize} \sum_{(u,v) \in E_S} \left(D(u,v) \sum_{(i,j) \in E_R} f_{(i,j),(u,v)} \right)$$

subject to the seven constraints defined above. The objective expresses the quality of an SFC-E solution as a summation of physical link delays, disregarding links that have not been utilized by the SFC. We note that if all physical links have the same propagation delay, then D can be removed from the objective function, which will then minimize the hop count of embeddings.

5.1.3.4 Single-agent RL optimization

Q-learning is a well-known model-free RL algorithm. It attempts to find optimal policies via a direct estimation of Q-values, which are often maintained in a tabular format. Each time a state-action pair is visited, the respective Q-value is updated. An apparent restriction here is the lack of generalization capacity. That is, in problems with large state-action spaces, the risk of scarce Q-updates is high.

To overcome this limitation, Mnih *et al.* [37] propose a Q-function approximation scheme based on deep neural networks (NNs), commonly termed deep Q-learning (DQL). As per DQL, the learning capacity of the agent lies in a NN which approximates the Q-function by parameterizing it, i.e., $Q(s,x) \sim Q(s,x;\theta)$ (where θ represents a vector of weights and biases of the NN). Ultimately, given a state as input, the NN computes a Q-value for every possible action. Then, it is up to a selection strategy (e.g., ε-greedy) to choose a particular action.

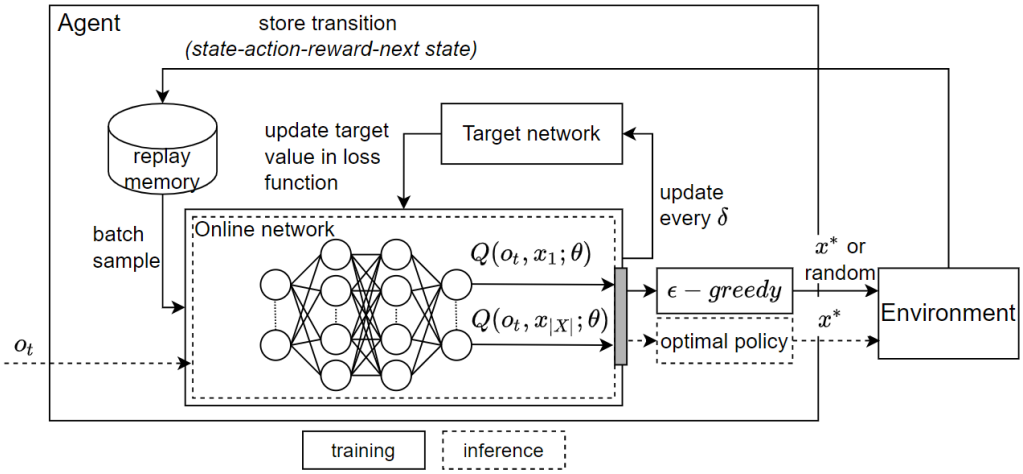

Figure 21. *Illustration of the typical DDQL architecture.*

Two mechanisms that contribute to the success of DQL are the replay memory and the coordination of an online NN and a target NN (see Van Hasselt *et al.* [38]). The replay memory is used to store state-action-reward-next state transitions; with proper sampling over it, we can subsequently train the online NN with uncorrelated data. The interplay of the two NNs works as follows: initially, the target NN is an exact replica of the online NN, meaning that they share the same architecture, weights and biases. However, it is only the online NN that is updated at every training step, while its weights and biases are copied to the target NN every δ training steps. The target NN (represented by θ^-) is used to temporarily stabilize the target value, which the online NN (represented by θ) tries to predict. Effectively, the training of the learning agent boils down to the minimization of the loss function. In the target term of the loss function, the greedy policy is evaluated by the online NN, whereas the greedy policy's value is evaluated by the target NN. As Van Hasselt et al., this reduces the overestimation of Q-values, which would be the case if both the greedy policy and its value had been evaluated by the online NN. An illustration of the DDQL framework is given in Figure 21. The loss function is given by

$$L(\theta_t) = E\left[\left(\underbrace{r_t + \gamma Q\left(s_{t+1}, arg\max_{x \in X} Q(s_{t+1}, x; \theta_t); \theta_t^-\right)}_{\text{target}} - \underbrace{Q(s_t, x_t; \theta_t)}_{\text{predicted}}\right)^2\right]$$

We devise a single-agent RL algorithm for SFC-E to serve as a baseline for comparison with our MARL scheme. In our implementation, a training episode consists of the placement of an entire SFC, whereas a decision step refers to the placement of a single VNF of the SFC. The RL framework consists of

- ***Observations.*** At time t, the agent observes the information o_t about the current VNF i and the SFC G_R it belongs to, as well as information about the available resources of the substrate's physical topology G_S. In particular, the agent receives the length of the SFC $|V_R|$, the coordinates of the src ($src.loc$) and dst ($dst.loc$) nodes of the SFC, the CPU demand of VNF i $d_R(i)$, the ingress $d_R(i^-, i)$ and egress $d_R(i, i^+)$ bandwidth requirements of VNF i, as well as its order $i.order$ in the SFC. Regarding the physical network, the agent observes the coordinates $u.loc$ and the available CPU $d_{S,t}(u)$ of each PoP $u \in V_S$, and the available bandwidth $d_{S,t}(u,v)$ of each link $(u,v) \in E_S$. Notice that, since o_t does not hold information about the servers of the topology (which are the elements that actually host VNFs), the environment is partially observable, hence we use observation o_t instead of state s_t.
- ***Actions.*** In our implementation, an action determines which PoP will host the current VNF i. Concretely, the agent's action set is $X = \{1, 2, \ldots, |V_S|\}$. Notice that if decision steps referred to the placement of an entire SFC instead of individual VNFs, then the action space would be substantially larger $|X| = |V_S|^{V_R}$, which would have an adverse effect on the algorithm's performance.
- ***Reward.*** A VNF placement is deemed successful if the selected PoP has at least one server with adequate resources to host it. An SFC placement (which is the ultimate goal) is deemed successful if all of its VNFs have been successfully placed and all of its virtual links are embedded onto physical links that connect the VNFs correctly. For every successful VNF placement, the agent receives r_t=0.1.

If the current VNF is the last VNF of the chain (i.e., terminal VNF) and is successfully placed, then the virtual link placement commences (using Dijkstra's shortest path algorithm for every adjacent VNF pair. If this process is successfully completed, the reward is computed as follows:

$$r_t = 10\frac{|opt_{path}| + 1}{|act_{path}| + 1}$$

where $|opt_{path}|$ is the length of the shortest path between the $src.loc$ and the $dst.loc$ (recall that these are always associated with PoP locations), and $|act_{path}|$ is the length of the actual path established by the DDQL algorithm. Apparently, the best reward r_t=10 is given when $|act_{path}| = |opt_{path}|$. In case the placement of any VNF or virtual link fails (because of inadequate physical resources), then r_t=-10 and a new training episode is initiated.

As hinted by previous discussions, the functionality of our DDQL agent relies on four key elements, namely an online NN, a target NN, a replay memory, and an action-selection strategy. Both NNs comprise an input layer whose size equals the length of the observation vector o_t (i.e., $o_t = 3|V_S| + |E_S| + 9$), two fully-connected hidden layers with 256 neurons each, and an output layer with $|V_S|$ neurons. All layers are feed-forward, the activation function applied on individual cells is Rectified Linear Unit (ReLU), and the loss function used is $L(\theta_t)$. The replay memory is implemented as a queue, which stores the latest 10000 environment transitions. The online NN samples 64 transition instances out of the replay memory at every training step, while the target NN is updated every $\delta = 20$ steps. Last, we employ an ε-greedy action-selection strategy $\varepsilon_t = \varepsilon_0(\varepsilon_{decay})^t$, where$\varepsilon_0$=1 and ε_{decay}=0.9998.

DDQL training consists of the following: At each episode, the agent handles the placement of a unique SFC. Prior to any action, the environment resets to an initial state. Then the first VNF of the current SFC is obtained, random loads for PoPs and physical links are generated, and the src and dst of this SFC are mapped to the respective PoPs. The core learning process is as follows. First, the agent chooses an action based on the current state, the environment transitions to a new state based on the action being taken, the transition is stored in the replay memory of the agent, the online NN is trained, and the current score and state are updated. Finally, when the environment transitions to the final state, during which the total reward is computed after the final VNF has been mapped.

5.1.3.5 Multi-Agent reinforcement learning

We now introduce cooperative multi-agent reinforcement learning (MARL) and discuss the independent Q-learning, which is the foundation of the proposed solution.

The simplest way of establishing a cooperative MARL framework is to treat each learning agent as an independent learning module. In this setting, if each agent is realized as a Q-learning agent, the respective MARL system is known as independent Q-learning (IQL) [39]. A major limitation of IQL is its lack of convergence guarantees, primarily stemming from the fact that the environment is non-stationary from the perspective of each agent. Effectively, this means that for an agent α, both its reward r_t and its next observation o_{t+1}^{α} are not solely conditioned on its current observation o_t^{α} and action x_t^{α}. In other words, the transitions of the environment and, by extension, the common rewards are affected by the actions of other agents as well. Although obtaining an optimized joint policy π while agents treat other agents as part of the environment is seemingly difficult, IQL exhibits good empirical performance [40].

Cooperative MARL is an active research field, given that numerous control tasks can be seen through the lens of multi-agent systems. In the context of collaborative DQL agents, recent works (e.g., [41] [42]) have established frameworks that promote joint training of multiple agents (i.e., centralized training - decentralized execution}), under the assumption that the Q^{total}-function of the multi-agent system can be decomposed into individual Q-functions such that, if each agent α maximizes its own Q^{α}, then Q^{total} is also maximized.

Irrespective of the promising advances in the field, our work builds upon typical IQL for the following reasons. First, IQL is simple enough to facilitate the interpretation of the system's learning behavior, as it avoids complex NN training schemes. Second, as discussed below (see Figure 22), we envisage an action coordination module at the NFVO layer to handle the constraints of SFC-E, which can significantly benefit the overall IQL framework.

We implement a DMARL scheme for SFC-E. Specifically, we generate a single DDQL agent for every PoP u in the substrate network G_S. For the rest of this subsection, we assume that agent α is associated with PoP u, and agent β with PoP v.

The MARL is defined by the following components:

- ***Observations***. At time t, each agent $\alpha \in A$ observes the information o_t^{α} about the current VNF i and the SFC G_R it belongs to, as well as information about the available resources of the substrate's physical topology G_S. In detail, agent α receives the length of the SFC, the coordinates of src and dst nodes of the SFC, the CPU demand of VNF i, the ingress and egress bandwidth requirements of

VNF i, and the order of this VNF in the SFC, similar to the single-agent observation described above. Regarding the physical network, agent α observes the coordinates of PoP u, the available CPU $d_{s,t}(u_s)$ of every server $s \in u$, and the available bandwidth of each physical link connected to PoP u. We further augment the observation space of agents by enabling cross-agent communications. Specifically, we define $N(\alpha)$ to be the set of neighboring agents of α, i.e., $N(\alpha) = \{\beta: distance(\alpha, \beta) = 1\}$, where $distance(\alpha, \beta)$ is defined as the length of the shortest path between PoPs u and v in G_S where agent α and β operate respectively. Effectively, agent α receives the location coordinates and the average CPU capacity from every PoP v managed by a neighboring agent $\beta \in N(\alpha)$. Cross-agent communications are pivotal for the efficiency of the system, since they enable agents to reason about the state of other agents. Yet, to maintain the communication overhead low, agents do not share their entire local state (i.e., the available CPU of each server), but rather a single descriptive value (i.e., the average available CPU across all servers). As explained earlier, agents within MARL settings operate over non-stationary environments, which practically implies that old experiences might become highly irrelevant as agents shift from exploratory to exploitative. To this end, we include a fingerprint [43] into the observation, namely ε the probability to select a random action, as a means to distinguish old from recent experiences. The last elements that are inserted into the observation o_t^α are three binary flags, namely $hosts_{another}$, $hosts_{previous}$, and $in_{shortest_{path}}$, which are computed by agent α prior to action selection. In particular, $hosts_{another} = 1$ if any VNF of the current SFC has been placed in PoP u, $hosts_{previous} = 1$ if the previous VNF of the current VNF has been placed in PoP u, and $in_{shortest_{path}} = 1$ if PoP u is part of the shortest path between the src and the dst of the SFC, while they are zero otherwise. These three flags will be proven crucial for analyzing the behavior of the DMARL system. Notice that, contrary to the single-agent case, here the team of agents observes the true state of the environment. However, from the point of view of a single agent, the environment is still partially observable.

- ***Actions***. Agents are equipped with a set of actions that allows them to express their intents based on their local state. In detail, all agents share the same discrete set of actions $X = \{0,1,2\}$, where 0 indicates low willingness, 1 expresses a neutral position, and 2 indicates high willingness with respect to hosting the current VNF. Further, the joint action $\{\boldsymbol{x}_t = \{x_t^1, \dots, x_t^n\}$ is forwarded to an action coordination module positioned at the NFVO layer, which treats individual actions as soft decisions and eventually computes the firm decision according to the overall objective (e.g., load balancing, consolidation) and constraints (e.g., each VNF at exactly one PoP). Here, we opt for a simple action coordination scheme where (i) the agent with the highest action (ties broken arbitrarily) will enable its associated PoP to be selected to host the current VNF, and (ii) the objective is solely dictated by the reward function, meaning that the coordinator does not intend to achieve another objective. Note that the above action sets are topology-agnostic (in contrast to the action set of single-agent DDQL, where there is one action for each PoP).
- ***Reward***. Our DMARL algorithm adopts a reward scheme similar to the one presented in the single-agent case. That is, for every successful VNF placement, all agents receive $r = 0.1$, and for every successful SFC placement, the common reward is computed with $r_t = 10\frac{|opt_{path}|+1}{|act_{path}|+1}$ as before. In case of a failed SFC-E attempt (inadequate physical resources), agents receive $r = -10$.

The proposed DMARL scheme is shown in Figure 22, which illustrates the mapping of the proposed elements and functionalities onto the NFV MANO framework. At the top, we depict an SFC request which is acknowledged to the NFVO via its north-bound interface (NBI) in step 0. During step 1, the NFVO forwards the VNF and the SFC states to the underlying VIMs, and we assume that each VIM is associated with a single PoP. Additionally, each VIM is equipped with a DDQL agent similar to a single agent DDQL, their only difference being the input and output layers. In particular, the input layer of each DDQL agent α in the DMARL setting has $|o_t^\alpha|$ neurons, while its output layer consists of three neurons (one for each action). The API calls depicted in step 2 implement the cross-agent communication functionality, where each agent retrieves condensed information from its neighbors. Step 3 refers to the computation of the local state, i.e., monitoring of the available CPU of each server and updating the three binary flag values. Afterward, agents convey their soft actions via the NBIs of their VIMs towards the south-bound interface (SBI) of the NFVO step 4), where individual actions are aggregated into a joint action that is handed over to the coordination module. The latter resolves potential conflicts and, in principle, assesses individual preferences with respect to the overall objective (step 5). Finally, the firm decision, which indicates the PoP that will host the current VNF, is sent to the respective VIM (step 6). This process is repeated until all VNFs and virtual links are placed or until the SFC-E fails.

The training procedure followed by the proposed DMARL scheme exhibits many similarities with the training of a single DDQL agent. Their core differences are as follows. First, the number of actions is no longer determined by the number of PoPs; it rather becomes a parameter of the multi-agent framework.

Specifically, the number of actions affects (i) the capacity of agents to express their resource allocation intents and (ii) the duration of the system's convergence. Another difference between the two algorithms lies in the manner in which the state is interpreted. In DMARL training, state refers to the VNF and the SFC state only, which is shared across all agents (see step 1 in Figure 22). However, further subroutines are called so that each agent chooses an action based on additional observations, such as its local state and neighboring (condensed) states (see steps 2 and 3 in Figure 22). Subsequently, the coordination module derives the firm action, and the environment transitions to a new state based on this action, as explained in step 4 in Figure 22. Each agent stores its transition into its (local) replay memory, and in all agents train their online NNs.

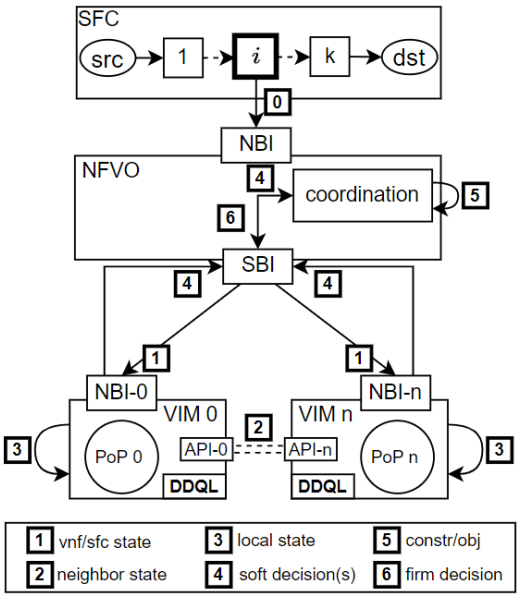

Figure 22. *The proposed DMARL scheme is decomposed from step 0 to step 6 and mapped to the NFV MANO framework.*

The simulation results will be reported in D5.2.

5.1.4 N-MAPE-K representation

Table 26 shows how the placement algorithm explained above maps on the N-MAPE-K framework.

Table 26. *Mapping the placement algorithm on the N-MAPE-K framework.*

N-MAPE-K module	Centralized single agent DDQL	Distributed MARL scheme + coordinator
Monitor	The centralized single agent DDQL uses the average of the load on the servers in the PoPs and their incoming and outgoing links; the requirements of the SFC to be embedded, the location of source and destination VNF.	The distributed MARL uses detailed information of the PoP is governing (i.e., the precise load on all its servers), but only averages of the loads on neighboring PoPs and no information of PoPs that are more than 1 hop away, besides the requirements of the SFC to be embedded. Furthermore, it uses some flags to convey information from 1n VNF mapping to the next.
Analyze	The analysis is implicit in the NN that takes the decisions.	
Plan	The single agent DDQL agent takes a direct decision where to allocate the SFC.	In the MARL scheme each agent first takes its own local soft decision whether it cannot, is neutral to or likes to host a VNF. These soft decisions subsequently need to be consolidated by a coordinator.
Execution	The (consolidation) decisions are sent to the PoPs who implement them.	
Knowledge	The knowledge resides in the synapses of the neural networks (which are tuned by optimizing the total reward)	

Training Loss / State – Actions – Rewards	Training reward reflects how well the SFC is embedded. DDQL employs a target and online NN to speed up convergence. Neither single agent DDQL nor MARL uses the true state as input as these would require too much bit rate overhead, rather they rely on observations that are summaries of state information. Actions determine how the SFC needs to be embedded. The reward reflects that successful SFC embeddings are rewarded a lot, unsuccessful ones are punished a lot. As VNFs of SFCs are getting embedded small bonuses are given to successful mappings of VNFs to further guide the learning process.

5.2 Auto scaling

Services, especially the ones consumed by humans, vary in popularity over time, i.e., active periods alternate with quiet periods. To fully exploit the flexibility of 5G and 6G, the amount of resources spent on each service should follow this trend in the demand for each service. Scaling algorithms are designed to follow the demand for each service by increasing the total amount of resources if the amount is insufficient to serve the demand or decreasing them in the opposite case.

While in deliverable D4.1 [3] we introduced the problem, here we propose two specific solutions: a traditional control-theoretic solution and RL based scaling solution. In D5.2, we will compare the performance of both.

5.2.1 Requirements in D2.2

The proposed scaling methods fulfill the following requirements proposed in D2.2 [1] as shown in Table 27.

Table 27. *Requirement satisfaction of the proposed scaling methods.*

Requirement from D2.2 [1]	Rationale
FR-MTERM-006	The scaling methods can be used by orchestrators to anticipate to sudden incoming traffic surges.
FR-MTERM-007	The scaling methods can reconfigure on-the-fly the number of replicas with minimal human intervention.
FR-SLMANO-003	The scaling methods converge to stable control of the number of replicas in a NFV environment.

5.2.2 Guidelines from D2.2

Using the Cart-Pole environment as inspiration and not a networking rationale, the definition of the reward function for the RL-based scaler was able to achieve stable control of the number of replicas in an NFV environment.

5.2.3 Solution description

In Section 6.2.1 of D4.1 [3] we introduced the autoscaling problem. In the context of 5G and 6G networks, scaling is a fundamental Management and Orchestration (MANO) operation in NFV that allows Network Operators (NOs) to automatically resize Virtualized Network Functions (VNFs) at runtime to meet traffic surges while providing performance guarantees.

Since scaling is considered a decision-making problem, as it determines the number of replicas that achieves Service Level Agreements (SLAs) among multiple stakeholders, it can be modeled as a Markov Decision Process (MDP). An MDP is a discrete-time stochastic framework for modeling decision-making problems. This process is defined by a tuple $(\mathcal{S}, \mathcal{A}, p, r)$, where $\mathcal{S}$ is a set of states, A is a set of actions, p is the transition probability between states s and s' after action a is taken, and r is the immediate reward obtained for performing action a. The policy, defined as π, is a mapping function from states to actions. The solution to an MDP is an optimal policy that maximizes the expected long-term reward (which is a discounted sum of immediate rewards).

Lately, Reinforcement Learning (RL) has been used to solve MDPs. The optimal policy is found in RL after many agent interactions with the environment (i.e., in a steady state). Q-Learning is the most used algorithm to find this optimal policy by defining a Q-function for all state-action pairs to measure how good the policy is. Therefore, finding the optimal policy is reduced to finding the action that maximizes this Q-function. Note that the state-action pairs can be stored in a table (i.e., Q-table); however, its size will grow substantially according to the number of states (e.g., a continuous state space), preventing its wide adoption. Instead, DQN uses a NN as a Q-function approximator to overcome the Q-Table's size limitation.

Contrary to most of the RL applications in networking, where the states, actions, and reward function are defined using a networking rationale, in [44] we map the auto-scaling problem to a known RL application. Specifically, the Gym Open-AI project provides a set of classical problems for RL algorithm benchmarking. We noticed that the scaling problem closely resembles the Cart-Pole[3] environment. In scaling, the agent tries to guarantee a given SLA by taking discrete actions (i.e., increase, decrease or maintain the number of VNF replicas). Similarly, in the Cart-Pole, the cart tries to keep the pole upright by taking discrete actions (i.e., go left or right). Following the same rationale as in the Cart-Pole environment, we define the information retrieved by the monitor as the network state. At time step t, the state $s(t)$ is defined as:

1. Mean CPU usage among the active VNFs.
2. Mean number of jobs waiting in the queue.
3. Peak (maximum) latency from the active VNFs.
4. The number of active VNFs.

Based on this information, the DQN agent decides if the number of VNF instances must be increased, decreased, or kept the same. The reward function is also defined in a similar way as in the Cart-Pole problem. Our agent takes discrete actions to maintain a given continuous variable (e.g., latency) at a certain level. Consequently, the agent is rewarded if the actions are leading towards that goal. More specifically, the reward function at time step t is defined as

$$r(t) = \begin{cases} 1 & |d(t) - d_{tgt}| < \epsilon \cdot d_{tgt} \bigvee |cpu(t) - cpu_{tgt}| < \epsilon \cdot cpu_{tgt} \\ 0 & |d(t) - d_{tgt}| \geq \epsilon \cdot d_{tgt} \bigvee |cpu(t) - cpu_{tgt}| \geq \epsilon \cdot cpu_{tgt} \\ -100 & \text{in episode termination cases} \end{cases}$$

where $d(t)$ is the peak latency from the active VNFs at time step t (taken from the network state), d_{tgt} is the target latency as defined by the SLA and ϵ is a range of tolerance (e.g., 20%). Notice that if the reward function is only defined based on the perceived latency, the agent will take the most obvious action: to keep increasing the number of instances, disregarding the economic impact of such a decision. To avoid the number of instances increasing without control, we also let the agent be rewarded if the current CPU usage is within a predefined range. If the CPU usage is low, probably the workload can be served using fewer VNFs and vice versa. Moreover, the agent is hardly penalized if it incurs in a wrong behavior, such as creating more replicas than necessary or exceeding the perceived latency beyond an allowed upper bound.

Alternatively, the scaling problem can also be solved using control theory. Here, the controller regulates a control variable, e.g., the number of resources, to keep a (measured) variable controlled, e.g., the KPI's specified in the SLA, as close as possible to a target value within some allowed boundaries. For instance, a Proportional–Integral (PI) controller calculates an error, which represents the deviation between the target and the measured value, and a trend, which captures how the measured value evolves. As the objective is to keep the measured variable close to the target, the correction signal, i.e., the change that needs to be applied to the control variable, is a weighted sum of the error and the trend. Moreover, if the system to be controlled is linear, the controller's parameters can be optimally set using Fourier or Laplace domain analysis. However, for non-linear systems, tuning the controller's parameters can be lengthy and computationally expensive. Among the advantages of using PI controllers for scaling is their steady response to small changes in the input variable (i.e., incoming traffic load).

In [44] we defined a PI-Controller that uses the current $d(t)$ and previous $d(t-1)$ peak latency to decide how to set the number of instances. In particular, it keeps track of a variable $\delta(t)$ at time step t:

$$\delta(t+1) = \delta(t) + \alpha\big(d(t) - d_{tgt}\big) + \beta\big(d(t) - d_{tgt}\big)$$

The second and third terms in the equation above are the integral and proportional terms, respectively. The former tries to keep the peak delay around the target, while the latter tries to proactively react to trends in the latency evolution. Notice that there is no differential term in the equation and that the PI-controller only needs the peak latency (current and previous value) as input. In particular, it does not need CPU usage.

If, at the beginning of time step $t+1$,

- $\delta(t+1) \geq 1$, then the number of replicas is increased by one and $\delta(t+1)$ is decreased by one,
- $\delta(t+1) \leq -1$, then the number of replicas is decreased by one and $\delta(t+1)$ is increased by one,
- $-1 < \delta(t+1) < 1$, then the number of replicas is kept the same.

[3] https://gym.openai.com/envs/CartPole-v1/

5.2.4 N-MAPE-K representation

The scaling methods defined above are mainly designed following a closed-loop control, so they can be easily mapped to the N-MAPE-K representation, as shown in Table 28.

Table 28. *N-MAPE-K decomposition of the scaling methods defined above.*

N-MAPE-K Component	Control theory-based scaler	RL-based scaler
Sensors + Monitor	Network State: service latency, end-to-end delay.	Resource State: CPU utilization, number of replicas Network State: service latency, end-to-end delay.
Analyze	Computation of how the control variable needs to be changed as a weighted sum of an error term and a trend.	The monitored variables are averaged to compose the State
Plan		A DQN agent takes the best action according to the learned strategy and current network state.
Execution + Effector	An API to the MANO platform to communicate scaling decisions.	
Knowledge	The control term's values.	Strategy with the actions to be taken according to the network state.
Training Loss / State – Actions – Rewards	N/A	States: Avg CPU utilization, Avg latency Rewards: Resource utilization tolerance.

6 Application of DAEMON NI-native architecture in WP4 solutions

With the goal of assessing the software architecture proposed in D2.2 [1] in the context of the solutions presented in WP4, in this section, we will discuss the expressiveness of the current version of the N-MAPE-K model to represent all kinds of interaction between WP4's solutions. We have identified three kinds of mismatches: (i) coordination between two different solutions that need to share some Knowledge (the K of the N-MAPE-K); (ii) NIFs sharing various N-MAPE-K components; (iii) cross-domain analysis of anomalies, which creates a clear need in terms of NI orchestration.

These and other possible requirements to be supported by N-MAPE-K will be addressed in WP2.

6.1 Coordination of energy-aware VNF placement and vRAN orchestration

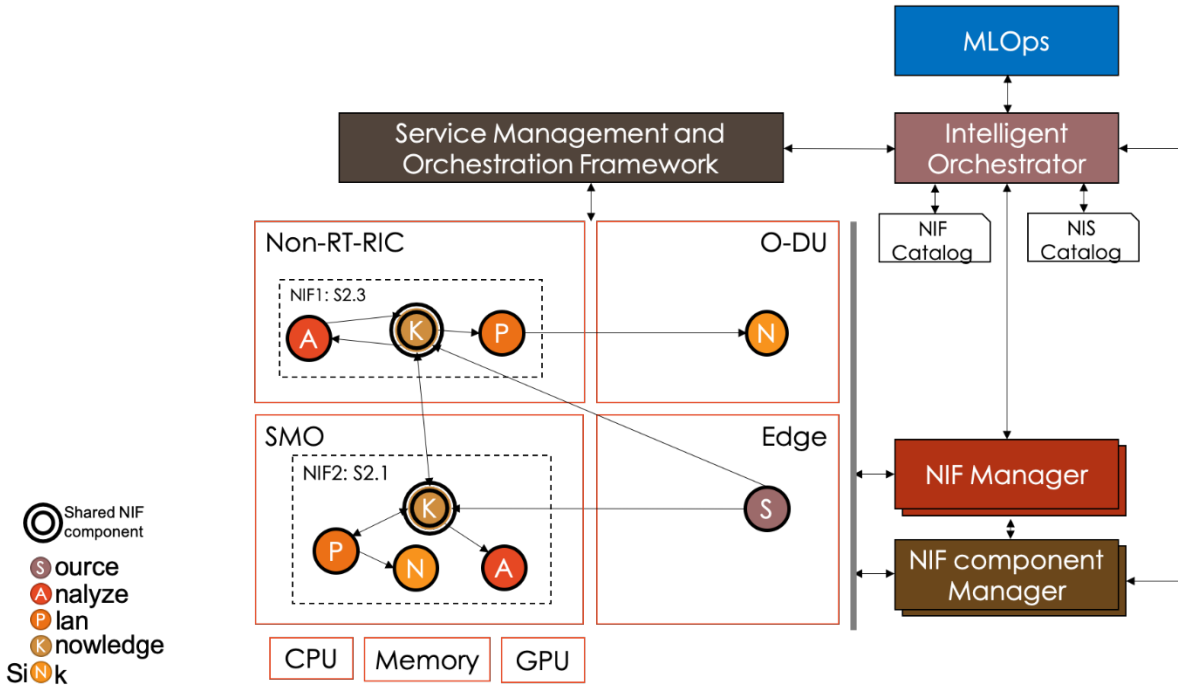

Figure 23. *Example scenario illustrating the need for coordination among NIFs.*

Figure 23 depicts a shared representation of two NI Functions: NIF1 (energy-driven vRAN orchestration, introduced in Section 2.3), and NIF2 (energy-aware VNF placement, introduced in Section 2.1). Illustrating a common representation of both NIFs in the context of the high-level architecture presented in D2.2 [1], allows us to identify additional requirements to enhance the architecture.

First, we can observe that some NIF Components (NIF-C) are shared among NIFs. DAEMON's NI plane has been specifically designed to support such a case. In the case of the two NIFs shown above, energy consumption measurements from an edge cloud platform are required and a source node component is hence shared.

Second, the above figure illustrates the need for coordination from the architecture, which is currently not supported. On the one hand, NIF1 generates knowledge about high-performing RAN control policies given a context and once virtualized instances of RAN components have been deployed. On the other hand, NIF2 is in charge of VNFs placement, which in this case implements virtualized RAN functions. This observation suggests that coordination between these two NIFs would be required, coordination that shall be provided in a centralized manner by the NI plane.

Such centralized coordination would allow sharing of knowledge that fostered synergetic performance improvements between both NIFs (sharing knowledge is represented by the double line of the knowledge NIF component in Figure 23). For instance, part of the knowledge learned by NIF1 can be used by NIF2 to make better placement decisions and, vice versa, some knowledge learned by NIF2 can be used by NIF1 to enforce informed (placement-aware) RAN control policies. Though such coordination is not available in the NI plane at this moment, this example of integrated NIFs does motivate the need for it to be studied in WP2.

6.2 Component sharing across capacity forecasting and anomaly detection

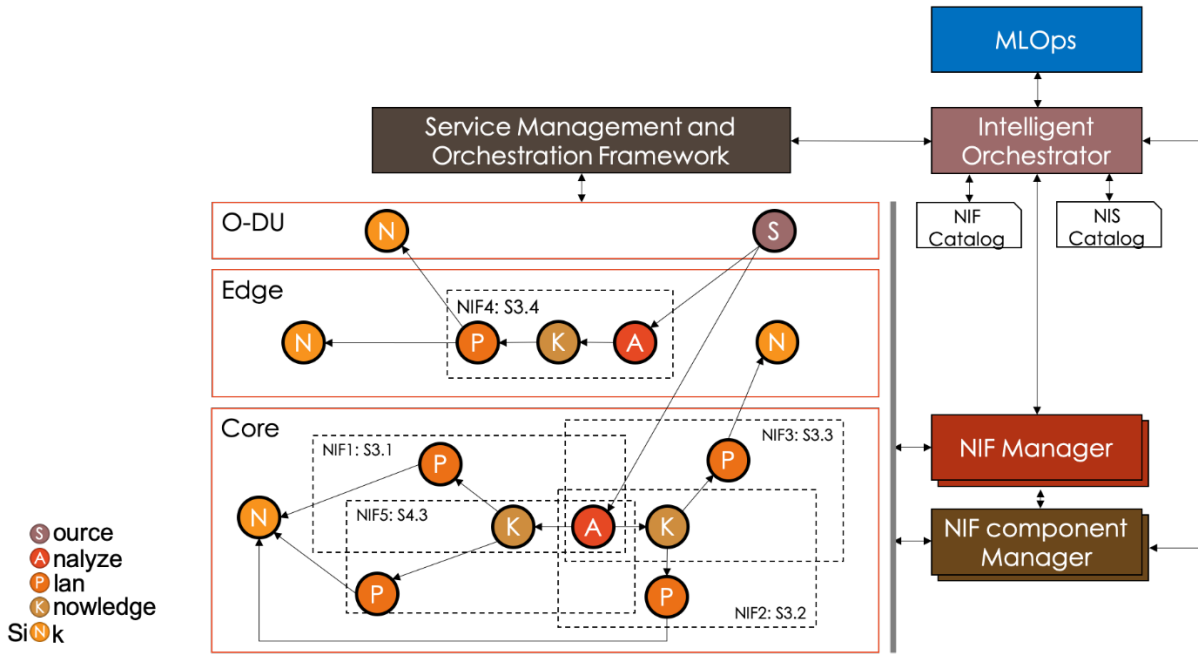

Figure 24. *Example scenario illustrating the need for component sharing mechanisms among NIFs.*

Figure 24 shows a representative situation where multiple NIFs share a substantial number of components, which calls for tight integration of the NIFs implementation as well as sensible management of their operation. Specifically, the example shows the N-MAPE-K representations of five different NIFs presented earlier in this document: (i) NIF1 is the preemptive capacity allocation solution presented in Section 3.1; (ii) NIF2 is the VM autoscaling solution presented in Section 3.2; (iii) NIF3 is the compute resource forecasting solution presented in Section 3.3; (iv) NIF 4 is the Edge caching and computing resource reservation solution presented in Section 3.4; (v) NIF 5 is the traffic volume anomaly prediction solution presented in Section 4.3.

The N-MAPE-K representations of these NIFs in the figure offer a clear view of their composition and interactions. We can observe how all solutions feed from information about mobile network traffic loads (expressed in byte/s of traffic) measured at O-DUs, hence close to the radio access, which is represented as a single Sink. NIF4 then fully resides in the network Edge, where the whole pipeline of NIF-C occurs, and actions are then enacted on resources at the Edge as well as in O-DUs.

The remaining NIFs have a much more entangled behavior. They all have Analyze stages that aim at preprocessing the O-DU-level data into traffic aggregates that are suitable for ingestion by machine learning models, implemented by their respective Plan stages. Yet, as all Analyze stages of NIF1, NIF2, NIF3 and NIF5 do exactly the same preprocessing, they can be mapped to a single component; instead Plan stages are different since each NIF has a specific goal in terms of decision making. Interestingly, it is also possible to pair the Knowledge stages of NIF1 and NIF5 (which both need to store historical data about traffic time series at O-DU level), as well as those of NIF2 and NIF3 (which both require storing past information about traffic aggregates at Edge datacenter level).

The figure also depicts how NIF1, NIF2 and NIF5 aim at acting on compute resources that are located in the Core network and that can potentially match a single Sink component. NIF3 aims instead at Edge resource management hence has a dedicated Sink stage (which is also different from that of NIF4 since the two NIFs have diverse targets in terms of management).

In a setting such as the one presented above, the presence of a NI plane appears paramount: a properly coordinated orchestration of NIFs and their component is required to ensure that monitoring and compute resources in the infrastructure are used in the best way possible, e.g., avoiding the unnecessary duplication of data collection and preprocessing tasks that are in fact identical many across NI instances.

6.3 Cross-domain NI orchestration for anomaly detection

The anomaly detection approach that we propose in Section 4.2. of D4.2 is meant for operators of international IoT platforms. This solution leverages signaling traffic between the mobile operators that connect the IoT devices (i.e., the SIM provider and the radio network provider). Because the IoT global interconnection model relies on international roaming, there are multiple entities involved in forming the end-to-end path from the IoT device to its application server. Thus, the data path traverses multiple

domains, which brings new challenges in terms of running root-cause analysis of anomalies, and further scaling towards the parties that are involved. For example, when detecting an anomaly in the connectivity of a cellular IoT device, the cause can lie with the interconnect, with the radio access provider, or with the SIM provider. ***All the parties involved in building the communication path for the IoT devices suffering from anomalies should be notified to further debug the connectivity issues***.

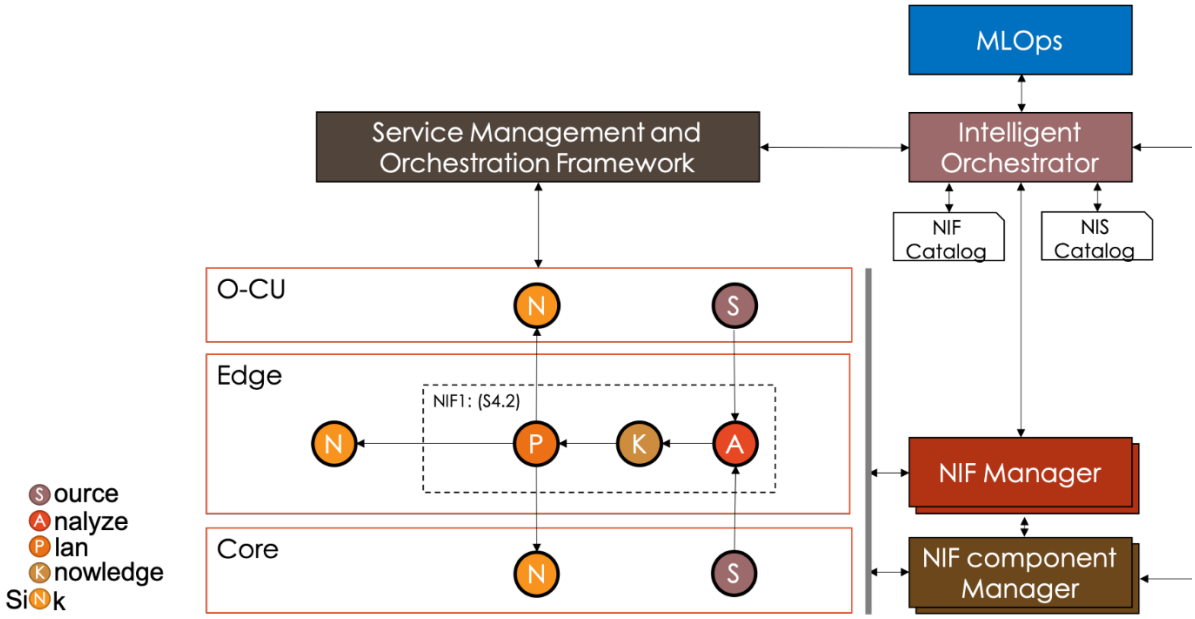

Figure 25. *Example scenario illustrating the need for cross-domain orchestration to implement one NIF.*

7 Conclusion and Outlook

Deliverable D4.2 presented the progress on the 13 NI solutions which were initially reported in D4.1 [3]. We discuss the developments for each Task that materialize ideas and satisfy the requirements that were identified in WP2. We start with solutions that include energy-aware VNF orchestration in open-source MANO infrastructure, proactive auto-scaling VNF control solutions, and energy-focused vRAN orchestration mechanisms for different application-specific performance KPIs. Moving to Task 4.2, we report our work on anticipatory capacity allocation with hybrid statistical-learning models, with methods that use meta-learned loss functions for reservation of virtual machines and slice reservations, and with our suggestion for optimistic learning techniques which combine offline and online learning with application to edge computing and caching. In the context of Task 4.3, we have developed federated learning techniques for anomaly detection in cellular networks, proactive anomaly detection solutions for large IoT systems, and anomaly forecasting solutions applied to traffic patterns of mobile services. And we also studied the noisy neighbor problem experimentally in virtualized computing platforms (O-RAN) from the perspective of anomaly detection. Finally, Task 4.4 studies Reinforcement Learning techniques for the placement and routing of service requests and control-theoretic approaches for automated scaling of resources that are assigned to services supported by the network. Both approaches target MANO-type solutions for Beyond-5G systems.

The above results are presented in a detailed fashion, using rigorous analytical models and/or data-driven (ML-based) techniques. Their development followed a close interaction across the different work packages, starting from extensive discussions towards satisfying (and revisiting, when necessary) the operational requirements stated in WP2 and including evaluations and assessments of these solutions in the context of WP5. Towards the preparation of the deliverable, the consortium held regular discussions and internal workshops for iterating among requirements the proposed solutions and their evaluation before finalizing the ideas presented in this document. We have also taken a critical approach during the development of these solutions, in line with the manifesto and mission of DAEMON, where we assessed the limitations of AI and proposed remedies when we identified such gaps. For instance, we have shown that the pre-selection of loss functions in capacity forecasting methods can severely harm performance, while, on the other hand, learning these functions (in a meta-learning-type of approach) yields much improved results, and we applied these ideas to different forecasting problems in B5G. As another example, we identified the issue of lack of (representative) training data which can lead to arbitrarily bad AI performance and proposed optimistic learning techniques which combine offline and online learning. This hybrid approach leads to accelerated learning whenever the AI model is valid and still maintains robust learning performance (accurate but slow) when the model fails.

The proposed solutions will be further developed in the remaining period to cover additional use cases, will be compared to alternative approaches, and will be evaluated using testbeds and new datasets. Towards the final milestones of this project, we will also focus on identifying and resolving potential conflicts of the proposed solutions for those operating in related decision spaces, and we will explore opportunities for synergies among them. For example, joint design of optimistic learning techniques and forecasting models. The results will be reported in D4.3, using the N-MAPE-K representation.

8 References

[1] ICT-52, "DAEMON Deliverable 2.2: Initial DAEMON Network Intelligence framework and toolsets," 2022.

[2] ICT-52, "DAEMON Deliverable 2.1: Initial report on requirements analysis and state-of-the-art frameworks and toolsets," 2021.

[3] ICT-52, "DAEMON Deliverable 4.1: Initial design of intelligent orchestration and management mechanisms," , November, 2021.," 2021.

[4] ETSI, "Open Source MANO (OSM) scope, functionality, operation and integration guidelines," 2019.

[5] A. Cañete, A. Rodríguez, M. Amor and L. Fuentes, "Energy-aware placement of network functions in edge-based infrastructures with Open Source MANO and Kubernetes," in *," in 3rd International Workshop on Architectures for Future Mobile Computing and Internet of Things*, Sevilla, 2022.

[6] K. Zhang, Y. Mao, S. Leng, Q. Zhao, L. Li, X. Peng, L. Pan and S. Maharjan, "Energy-Efficient Offloading for Mobile Edge Computing in 5G Heterogeneous Networks," *IEEE Access,* vol. 4, pp. 5896-5907, 2016.

[7] Y. Mao, C. You, J. Zhang, K. Huang and K. B. Letaief, "A Survey on Mobile Edge Computing: The Communication Perspective," *IEEE Communications Surveys & Tutorials,* vol. 19, no. 4, pp. 2322-2358, 2017.

[8] L. de Moura and N. Bjørner, "Z3: An Efficient SMT Solver," in *Tools and Algorithms for the Construction and Analysis of Systems.*, vol. 4963, C. R. J. Ramakrishnan, Ed., Lecture Notes in Computer Science,, 2008.

[9] R. Wimmer, N. Jansen, E. Ábrahám, B. Becker and J. Katoen, "Minimal Critical Subsystems for Discrete-Time Markov Models," in *Tools and Algorithms for the Construction and Analysis of Systems.*, vol. LNCS 7214, 2012.

[10] M. Ganeshalingam, A. Shehabi and L.-B. Desroche, "Shining a Light on Small Data Centers in the U.S.," 2017.

[11] M. Abdullah, W. Iqbal, A. Mahmood, F. Bukhari and A. Erradi, "Predictive Autoscaling of Microservices Hosted in Fog Microdata Center," *IEEE Systems Journal,* vol. 15, no. 1, pp. 1275-1286, 2021.

[12] T. Chen, R. Bahsoon and X. Yao, "A Survey and Taxonomy of Self-Aware and Self-Adaptive Cloud Autoscaling Systems," *ACM Comput. Surv.*, vol. 51, no. 3, p. 40, 2018.

[13] Mohammad S. Aslanpour, Sukhpal Singh Gill, Adel N. Toosi, "Performance evaluation metrics for cloud, fog and edge computing: A review, taxonomy, benchmarks and standards for future research," *Internet of Things,* vol. 12, p. 100273, 2020.

[14] A. Cañete, K. Djemame, M. Amor, L. Fuentes and A. Aljulayfi, "A proactive energy-aware auto-scaling solution for edge-based infrastructures," in *11th International Workshop on Cloud and Edge Computing, and Applications Management (CloudAM 2022)*, 2022.

[15] C. Sonmez, A. Ozgovde and C. Ersoy, "EdgeCloudSim: An environment for performance evaluation of Edge Computing systems," in *Second International Conference on Fog and Mobile Edge Computing (FMEC)*, 2017.

[16] S. Chen, L. Jiao, F. Liu and L. Wang, "EdgeDR: An Online Mechanism Design for Demand Response in Edge Clouds," *IEEE Transactions on Parallel and Distributed Systems,* vol. 33, no. 2, pp. 343-358, 2022.

[17] A. Cañete, M. Amor and L. Fuentes, "Supporting IoT applications deployment on edge-based infrastructures using multi-layer feature models," *Journal of Systems and Software,* vol. 183, pp. 111086,, 2022.

[18] Y. Sui and et al., "Safe exploration for optimization with Gaussian processes," in *International conference on machine learning. PMLR*, 2015.

[19] L. Lo Schiavo, M. Fiore, M. Gramaglia, A. Banchs and X. Costa-Perez, "Forecasting for Network Management with Joint Statistical Modelling and Machine Learning," in *," 2022 IEEE 23rd International Symposium on a World of Wireless, Mobile and Multimedia Networks*, 2022.

[20] A. Collet, A. Banch y M. Fiore, «LossLeaP: Learning to Predict for Intent-Based Networking,» de *IEEE INFOCOM 2022 -IEEE Conference on Computer Communications*, 2022.

[21] ICT-52, «DAEMON Deliverable 5.1: Preliminary Evaluation Results and Plan for Proof-of-Concept Demonstrations,,» 2022.

[22] P. Rousseeuw y M. Hubert, «Anomaly detection by robust statistics,» *WIREs Data Mining and Knowledge Discovery,* vol. 8, nº 2, 2018.

[23] Y. Gal y Z. Ghahramani, «Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning,» de *Proc. of The 33rd International Conference on Machine Learning (ICML)*, New York, NY, USA, 2016.

[24] ETSI, «Network Functions Virtualisation (NFV); Management and Orchestration. GS NFV-MAN 001 v1.1.1,» 2014.

[25] M. Rost and S. Schmid, " On the hardness and inapproximability of virtual network embeddings," *IEEE/ACM Transactions on Networking,* vol. 28, no. 2, p. 791–803, 2020.

[26] D. Dietrich, A. Abujoda, A. Rizk y P. Papadimitriou, «Multi-provider service chain embedding with nestor,» *IEEE Transactions on Network and Service Management,* vol. 14, nº 1, pp. 91-105, 2017.

[27] A. Pentelas, G. Papathanail, I. Fotoglou y P. Papadimitriou, «Network service embedding across multiple resource dimensions,» *IEEE Transactions on Network and Service Management,* vol. 18, nº 1, p. 209–223, 2020.

[28] S. Haeri y L. Trajkovi´c, «Virtual network embedding via monte carlo tree search,» *IEEE transactions on cybernetics,* vol. 48, nº 2, pp. 510-521, 2017.

[29] Y. Xiao, Q. Zhang, F. Liu, J. Wang, M. Zhao, Z. Zhang y J. Zhang, «Nfvdeep: Adaptive online service function chain deployment with deep reinforcement learning,,» de *Proceedings of the International Symposium on Quality of Service*, 2019.

[30] P. T. A. Quang, Y. Hadjadj-Aoul y A. Outtagarts, «A deep reinforcement learning approach for vnf forwarding graph embedding,» *IEEE Transactions on Network and Service Management,* vol. 16, nº 4, pp. 1318-1331, 2019.

[31] J. Pei, P. Hong, M. Pan, J. Liu y J. Zhou, «Optimal vnf placement via deep reinforcement learning in sdn/nfv-enabled networks,» *IEEE Journal on Selected Areas in Communications,* vol. 38, nº 2, pp. 263-278, 2019.

[32] J. Zheng, C. Tian, H. Dai, Q. Ma, W. Zhang, G. Chen y G. Zhang, «Optimizing nfv chain deployment in software-defined cellular core,» *IEEE Journal on Selected Areas in Communications,* vol. 38, nº 2, pp. 248–262,, 2019.

[33] R. Solozabal, J. Ceberio, A. Sanchoyerto, L. Zabala, B. Blanco y F. Liberal, «Virtual network function placement optimization with deep reinforcement learning,",» *IEEE Journal on Selected Areas in Communications,* vol. 38, nº 2, p. 292–303, 2019.

[34] Z. Yan, J. Ge, Y. Wu, L. Li y T. Li, «Automatic virtual network embedding: A deep reinforcement learning approach with graph convolutional networks,» *IEEE Journal on Selected Areas in Communications,* vol. 38, nº 6, pp. 1040-1057, 2020.

[35] L. Wang, W. Mao, J. Zhao y Y. Xu, «Ddqp: A double deep q-learning approach to online fault-tolerant sfc placement,» *IEEE Transactions on Network and Service Management,* vol. 18, nº 1, p. 118–132, 2021.

[36] J. Jia, L. Yang y J. Cao, «Reliability-aware dynamic service chain scheduling in 5g networks based on reinforcement learning,» de *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*, 2021.

[37] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, M. J. Veness, G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski y e. al., «Human-level control through deep reinforcement learning,» *Nature,* vol. 518, nº 7540, pp. 529–533,, 2015.

[38] H. V. Hasselt, A. Guez y D. Silver, «Deep reinforcement learning with double q-learning,» *Proceedings of the AAAI conference on artificial intelligence,* vol. 30, nº 1, 2016.

[39] M. Tan, «Multi-agent reinforcement learning: Independent vs. cooperative agents,» de *Proceedings of the tenth international conference on machine learning*, 1993.

[40] L. Matignon, G. J. Laurent y N. L. Fort-Piat, «Independent reinforcement learners in cooperative Markov games: a survey regarding coordination problems,» *" The Knowledge Engineering Review,* vol. 27, nº 1, pp. 1-31, 2012.

[41] G. L. A. G. W. M. C. P. Sunehag, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls y e. al., «Value-decomposition networks for cooperative multi-agent learning,» *arXiv preprint arXiv:1706.05296,* 2017.

[42] T. Rashid, M. Samvelyan, C. Schroeder, G. Farquhar, J. Foerster y S. Whiteson, «Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning,» de *International Conference on Machine Learning. PMLR*, 2018.

[43] N. N. G. F. J. Foerster, T. Afouras, P. H. Torr, P. Kohli y S. Whiteson, «Stabilising experience replay for deep multi-agent reinforcement learning,» de *International conference on machine learning. PMLR*, 2017.

[44] P. Soto, D. De Vleeschauwer, M. Camelo, Y. De Bock, K. De Schepper, C.-Y. Chang, P. Hellinckx, J. F. Botero y S. Latré, «Towards autonomous VNF auto-scaling using deep reinforcement learning,» de *2021 Eighth International Conference on Software Defined Systems (SDS)*, 2021.