H2020 DAEMON Project
Grant Agreement No. 101017109

Deliverable 5.1

Preliminary evaluation results and plan for proof-of-concept demonstrations

Abstract

This document presents the current status of DAEMON WP5 activities. It presents the validation activities, performance evaluations and functional assessment tests of all the NI-solutions developed in WP3 and WP4 to date. The performance evaluation aims at meeting 9 target KPIs in well-defined scenarios, which are measured and assessed by developing 7 evaluations. Each evaluation relies on a set of technical tools, employed to perform 23 experimental activities. Such activities target different facets of a single evaluation.

Document properties

Document number	D5.1
Document title	Preliminary evaluation results and plan for proof-of-concept demonstrations
Document responsible	IMDEA
Document editor	Michele Gucciardo (IMDEA)
Editorial team	Marco Fiore (IMDEA) Gines Garcia Avilés (i2CAT) Michele Gucciardo (IMDEA) Antonio Bazco Nogueras (IMDEA) Gabriele Baldoni (ADLINK) Ivan Paez (ADLINK) Inmaculada Ayala (UMA) Danny De Vleeschauwer (NBL) Chia-Yu Chang (NBL) Paola Soto (IMEC) Nina Slamnik (IMEC) Miguel Camelo (IMEC) Michail Kalntis (TUD) Marco Gramaglia (UC3M) Evangelos Kosmatos (WINGS) Ioannis Chondroulis (WINGS) Sokratis Mparmpounakis (WINGS) Ioannis-Prodromos Belikaidis (WINGS) Theodoros Kasidakis (WINGS) Panagiotis Demestichas (WINGS) To be completed – person D (partner acronym)
Target dissemination level	Public
Status of the document	Final
Version	1.0

Production properties

Reviewers	Marco Fiore (IMDEA) Evangelos Kosmatos (WINGS) Andres Garcia Saavedra (NEC)

Document history

Revision	Date	Issued by	Description
0.1	15/02/22	All partners	Initial content
0.2	01/03/22	Editor	First version, ready to be reviewed
0.3	15/03/22	All partners	Updated content based on feedback of editor and Marco Fiore
0.4	24/03/22	External reviewers	Version reviewed outside WP5
1.0	31/03/22	Editor	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

4G – Fourth Generation
5G – Fifth Generation
AI – Artificial Intelligence
ASIC – Application-Specific Integrated Circuit
AUC – Area Under Curve
B5G – Beyond 5th Generation
BBU – Baseband Unit
BNN – Binarized Neural Network
BS – Base Station
C-ITS - Cooperative Intelligent Transportation System
CNN – Convolutional Neural Network
CORD – Central Office Rearchitected as Data Center
CUPS – Control-User Plane Separation
DoA – Description of Action
DCB – Dynamic Channel Bonding
DL – Downlink
DNN – Deep Neural Network
DT – Decision Tree
EPC – Evolved Packet Core
eNB – Evolved Node-B
FNN – Feedforward Neural Network
FPGA – Field Programmable Gateway Array
GA+JCC - General Algorithm with Joint Cache Rental and File Caching
GB – Gradient Boost
gNB – Next Generation Node-B
GNN – Graph Neural Network
GPU – Graphical Processing Unit
IPC – Inter-Process Communication
KPIs – Key Performance Indicators
LTE – Long-Term Evolution
MAC – Medium Access Control
MANO – Management and Orchestration
MCU – Micro-Controller Unit
MEC – Multi-access Edge Computing
ML – Machine Learning
MNO – Mobile Network Operator
MOS – Mean Opinion Score
Near-RT – Near Real-Time
Non-RT – Non Real-Time
NFV – Network Function Virtualization
NFVI – Network Function Virtualization Infrastructure
NI – Network Intelligence
NR – New Radio
NSA – Non-Standalone
NS – Network Service
OBU – On-board Unit
OPEX – Operation Expenditure
OSM – Open-Source MANO
PHY – Physical layer
PoC – Proof-of-Concept
QoE – Quality of Experience
R-CNN – Region-based Convolutional Neural Network
RAN – Radio Access Network
RBF – Radial Basis Function
RCA – Root Cause Analysis
RIC – RAN Intelligent Controller
RIS – Reconfigurable Intelligent Surfaces
RMSE – Root Mean-Squared Error
RRC – Radio Resource Control
RSUs – Road-Side Units
RF – Rando Forest
RFR – Radio Frequency
RT – Real Time
RU – Radio Unit
SA – Standalone
SDR – Software Defined Radio
SLA – Service-Level Agreement
SMA – SubMiniature version A
SNR – Signal-to-Noise Ratio
SVR – Support Vector Regression
TCP – Transmission Control Protocol
ToR – Top of rack
TRL – Technology Readiness Level
UE – User Equipment
UL - Uplink

URLLC – Ultra-Reliable Low-Latency Communications
V2X – Vehicular-to-Everything
VM – Virtual Machine
VNF – Virtual Network Function
vBS – virtual Base Station
vRAN – virtualized Radio Access Network
WP – Work Package

Executive summary

This deliverable presents DAEMON's preliminary evaluation results concerning some preliminary NI-solutions developed in WP3 and WP4, targeting 8 network functionalities assisted via NI.

Specifically, WP5 receives the solutions for such functionalities with the goal of assessing them. The evaluation methodology employed by WP5, makes use of a set of tools that includes experimental testbeds, simulators or emulators, which are possibly fed with measurement data. Furthermore, WP5 is responsible of the implementation of the solutions from WP3 and WP4 into such experimental, simulation or emulation systems. Finally, WP5 provides feedback to WP2, WP3 and WP4 concerning the efficiency of both the design and the operation of the NI solutions developed within the project. According to the project timeline, such feedback occurs in three iterations of NI design/application/evaluation.

In this document, we present the results of WP5 activities in the very first of the three planned iterations. Therefore, this document reports preliminary outcomes about the effectiveness of the NI algorithms developed in the project to date, whose goal is also helping to improve NI design in WP2 and application in WP3 and WP4. The document consists of five sections.

In Section 1, we describe the structure of our performance evaluation methodology. We introduce the target KPIs defined to validate the NI-solutions, the evaluation methodology to measure such KPIs, and the tools exploited to perform tests in the different experimental activities.

In Section 2, we detail the 9 target KPIs to assess the performance, reliability and sustainability of these NI-solutions. We also describe 7 evaluations planned designed to prove the validity of the proposed techniques in realistic experimental or measurement data-driven settings.

In Section 3, we illustrate the complete set of technical tools, employed during the performance tests, which we categorized into experimental testbeds, simulators and emulators, and datasets.

In Section 4, we show the results of 23 experimental activities carried on by the consortium, each targeting different facets of individual evaluations.

In Section 5, we provide some conclusions and a final outlook of the document.

1 Introduction

Work Package (WP) 5 of the DAEMON project lays out all validation, performance evaluation and functional assessment tests of the solutions developed in WP3 and WP4. As illustrated in Figure 1, WP5 receives the solutions developed in WP3 and WP4 for network functionalities that we assist and automate via Network Intelligence (NI), and performs a comprehensive assessment of such solutions. The evaluation carried out by WP5 uses diverse tools for each functionality, including experimental testbeds, simulators or emulators, possibly fed with measurement data. Therefore, WP5 also takes care of implementing the solutions from WP3 and WP4 into such experimental, simulation or emulation systems.

As also shown in Figure 1, WP5 provides feedback to WP2, WP3 and WP4 about the efficiency of the design and operation of the NI algorithms developed in the project. According to the project timeline, this occurs in three iterations of NI design/application/evaluation. Here, we present the results of WP5 activities in the very first iteration above: therefore, ***this document reports preliminary outcomes about the effectiveness of the NI algorithms developed in the project to date, whose goal is also helping to improve the NI design in WP2 and application in WP3 and WP4***.

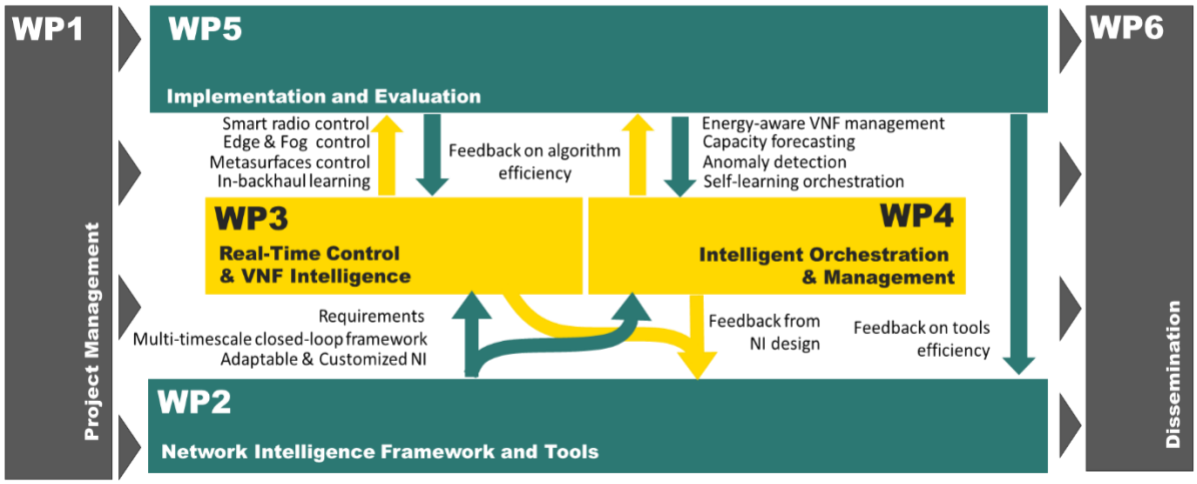


Figure 1. *Positioning of WP5 in the DAEMON project work plan.*

More precisely, and according to what stated in the Description of Action (DoA), ***the solutions developed in WP3 and WP4, and thus evaluated in WP5, target 8 key NI-assisted network functionalities***. Such functionalities are distributed across different micro-domains of the next-generation mobile architecture (namely, Core, Transport, Edge, Far Edge, and Beyond Edge), and across controllers, orchestrators and functions that operate at different timescales. They are listed as follows.

- Reconfigurable Intelligent Surfaces (RIS) control ;
- Multi-timescale edge resource management;
- In-backhaul support for service intelligence;
- Compute-aware radio scheduling;
- Energy-aware Virtual Network Function (VNF) orchestration;
- Self-learning Management and Orchestration (MANO);
- Capacity forecasting; and
- Automated anomaly response.

In this deliverable, we report the current results of different performance evaluations across the 8 NI-assisted solutions introduced above. Overall, the content of this document provides initial objective evidence of the advantage of a structured, deep, and sensible integration of NI into network infrastructures, and demonstrates the viability and performance of the NI-native vision for Beyond the fifth Generation (B5G) networks set forth by the DAEMON project.

In order to organize our performance evaluation in a structured way, ***we aim at meeting clear targets on a comprehensive range of Key Performance Indicators (KPIs)*** that illustrate the performance, reliability and sustainability of the techniques proposed in the project to date. We remark that these targets are clealy to be intended as limited to (i) the scope of the 8 specific functionalities developed in the project and (ii) the evaluation scenarios where such functionalities are deployed as part of our tests; indeed, we cannot make claims on performance beyond what we actually assess. The KPI targets are outlined in Section 2.1.

We measure the KPIs identified and assess whether the targets are satisfied by developing a complete set of 7 evaluations that are designed to prove the feasibility of the proposed NI solutions in realistic experimental or measurement data-driven settings. The list of evaluations is provided in Section 2.2, and relies on a comprehensive methodology based on the following complementary approaches.

i. *Real-world experiments.* Implementations in experimental testbeds are the primary option for the validation of the NI solutions. Within the project, ***we develop and/or take advantage of 10 experimental sites*** that feature cutting-edge research infrastructure and cover all network micro-domains. Each site has specificities that make it especially suitable to investigate precise subsets of the NI-assisted functionalities targeted by the project. ***The project evaluation sites allow for credible Proofs-of-Concepts (PoC) in realistic but controlled environments***, which showcase how Network Intelligence (NI) can drive zero-touch network management to yield substantial performance gains and savings in resource usage efficiency or energy consumption. The experimental sites used for real-world NI assessment are described in Section 3.1.

ii. *Data-driven evaluations.* We leverage realistic datasets to feed simulations or emulations that provide data-driven performance assessments of the NI solutions. ***We consider both large-scale traffic measurements*** (e.g., collected in nationwide operational mobile networks), ***as well as small-scale datasets*** (e.g., recorded in platforms deployed in laboratory environments). Such real-world data allows validating the proposed NI instance orchestration and NI-assisted functionalities in dependable settings, and possibly at scales that cannot be achieved with real-world experiments. By using substantial volumes of mobile traffic data, which is paramount to the proper training of the NI algorithms, we ensure that such algorithms are trained and tested in realistic conditions – ultimately supporting that the observed NI performance are aligned with those that could be expected in production systems. These evaluations do not solely rely on baseline testing of machine learning solutions for NI, but ***we also feed the measurement data to digital-twins implemented into simulation and emulation sandboxes***, including proprietary tools for real-time emulation developed by the project partners. In these cases, we follow a DevOps approach, using data-driven models and micro-services architectures that capture system dynamics at short timescales, so as to understand the scalability properties of the NI in controlled environments. The simulation and emulation tools developed and/or employed in the project are described in Section 3.2, whereas the datasets feeding them are described in Section 3.3.

The performance tests executed in the project to date, which employ the aforementioned tools, have been structured into a wide range of 23 activities, each targeting a specific and focused technical problem within scope of one of the 8 network functionalities addressed by WP3 and WP4. Specifcally, multiple activities address different facets of each individual evaluation, and can be combined so as to complete the whole set of target evaluations. The results of each activity are reported in Section 4.

Table 1. *List of performed evaluations to date, with the associated activities, tools and target KPIs.*

Evaluation	Short description of the evaluation	Planned KPIs	Tools used	Activities
E1	NI for sustainable virtualized Radio Access Network (RAN)	K1, K2, K4, K5, K9, K3, K8	T1, S2, T5	A1, A2, A3, A4, A5
E2	NI for VNF placement and control	K1, K2, K3, K8, K4, K5, K9, K7	T8, D10	A6, A7, A8
E3	NI for real-time anomaly detection	K3, K7, K4, K5, K8	T8	A9
E4	NI for Edge orchestration	K1, K2, K3, K4, K5, K8, K9, K7	D5, T4, D6, D1, S4, S1, D9	A10, A11, A12, A13, A14, A15, A16, A17
E5	NI for automated anomaly response	K3, K7, K5, K8, K9	D7, D8, D11, D12, D13, D14	A18, A19
E6	NI for capacity forecasting and self-learning	K2, K4, K9, K5, K6	D1	A20, A21, A22
E7	NI to configure a Reconfigurable Intelligent Surface	K2, K4, K9, K5, K6	T9	A23

As a summary, the evaluations developed in the project aim at achieving the KPI targets in controlled, yet relevant, environments. To do so, each evaluation relies on a set of technical tools (i.e., simulators and emulators, experimental testbeds, or datasets) that are employed to perform a number of detailed activities. ***Table 1 provides a complete view of all relationships among evaluations, KPIs and activitie***s: for each evaluation, in each row, we indicate the associated KPIs, the used tools and the list of activities. Table 1 is thus intended as a reference that assists the reader and helps following the organization of the performance evaluation of the project.

The detailed description of KPIs, evaluations, tools and activities is provided in the remainder of the document, which is structured as follows.

- *Section 2* details the target KPIs set for the NI solutions developed in the project, and the different evaluations that have been foreseen to validate such solutions.
- *Section 3* lists the technical tools that have been employed to implement the evaluations above, by assessing the performance of the NI solutions in realistic experimental or data-driven settings.
- *Section 4* shows the final results of the evaluations carried out to date in the project, which provide initial validations to the KPI targets through the different tools presented before.

2 Target KPIs and evaluations

The ambitious vision and objectives of DAEMON call for a credible and solid validation and performance evaluation of the eight NI-assisted functionalities introduced in WP3 and WP4 to improve B5G networks performance, sustainability and reliability. To date, we developed preliminary NI solutions, which:

- ***Increase network performance end efficiency***, by supporting network functionalities that allow for extremely fast and adaptive control of (i) RIS in a new Beyond Edge domain, (ii) computational resources in the Edge domain, and (iii) support for service intelligence in the user plane of the Transport domain.
- ***Enable the sustainable operation of B5G systems***, by means of solutions for (iv) computation-aware radio scheduling, and (v) energy-aware VNF orchestration and (vi) self-learning MANO.
- ***Ensure an extreme reliability of zero-touch B5G***, *by* exploiting (vii) capacity failure avoidance via anticipatory allocation, and (viii) anomaly detection.

2.1 Target KPIs

To evaluate the performance of all the NI-assisted functionalities in the project, we introduce nine KPIs accompanied by target values that we want to meet in controlled, but relevant, scenarios. ***Table 2 lists the KPIs fixed by 5G-PPP [1] and additional technical KPIs originally proposed by the project, with their respective targets.*** It also shows evidence supporting the feasibility of the targets and our progress towards achieving those targets. As anticipated in Section 1, these targets and the evidence outlined in Table 2 are to be intended as limited to (i) the scope of the 8 specific functionalities developed in the project and (ii) the evaluation scenarios where such functionalities are deployed as part of our tests; clearly, we cannot make claims on performance beyond what we actually assess.

More in detail: K3, K6, K8 and K9 are *performance KPIs*; K1, K2 and K4 are *sustainability KPIs*; K5 and K7 are *reliability KPIs*. The last column of Table 2 shows our progress achieving the assigned target for each KPI: such a value is a qualitative indicator calculated, for each KPI, computed as the average of the progress of all the experimental activities targeting that specific KPI. The detailed progress of the individual activities that is informing such a global per-KPI progress will be presented later in Table 10.

Table 2. *List of target KPIs: K1, K2, K4, K6 set the target relative to the current baseline; K3, K5, K7, K8, K9 define absolute targets that involve substantial improvements over today's state-of-the-art technology.*

KPI	Description	Target	Evidence supporting the feasibility of the target and improvement over the baseline	Average overall progress of the assocated activities
K1	VNF energy consumption reduction	50%	According to recent assessments, energy consumption in the edge and core of softwarized mobile networks may increase as much as 25% due to the impact of active cooling among other issues [2], [3]. Thus, DAEMON aims at saving of up to 25% of energy costs thanks to a NI-assisted VNF placement based on energy considerations. Furthermore, additional 25% savings will be allowed by NI-assisted VNFs that can adapt their energy footprint to the context of the location where they are running [4].	27%
K2	Saving of computational resources at the edge	40%	Current results of the DAEMON partners show that by applying intelligent radio and CPU scheduling in O-RAN architectures, one can reduce the requirement of the computing resources required by virtualized base stations by up to 20% with minimal impact on performance [5], [6]. The improved NI design developed by DAEMON shall advance those techniques from multiple perspectives as outlined in Objectives 1-3, hence making a 40% reduction target viable in the scenarios considered here.	37%
K3	Response time of AI-based NI algorithms	1 ms	By leveraging on recent advances on highly elastic Artificial Intelligence (AI) models [7], DAEMON will build NI algorithms capable of meeting the hard requirements of delay- and reliability-sensitive traffic through effective trade-offs with accuracy. To the best of our knowledge, DAEMON will provide the first AI-based NI with guarantees in terms of response time.	24%
K4	OPEX savings	60%	Preliminary studies of the DAEMON consortium demonstrate how AI models trained with customized loss functions that reflect monetary costs can avoid	41%

			Service-Level Agreement (SLA) violations and reduce Operation Expenditure (OPEX) by up to 40% [8]. While these figures refer to local solutions, the structured coordination of NI instances enabled by the DAEMON architecture will allow targeting further cost savings of up to 60%.	
K5	Reliability	Five 9's	Current state-of-the-art solutions developed by the DAEMON partners can satisfy SLA with a level of reliability between three and four 9's [9]. The cooperative, multi-timescale NI orchestration model envisaged in DAEMON will significantly improve the amount and quality of information available to NI algorithms, making it possible to target further gains in the reliability of resource allocation decisions to, e.g., meet Ultra-Reliable Low-Latency Communication (URLLC) requirements in certain conditions.	42%
K6	Wireless capacity (bps/m2) increase	100%	NI-controlled intelligent surfaces that can adapt the propagation properties of wireless channels to the environment dynamics will allow DAEMON to, at least, double the bit-rate per square meter in scenarios of interest, in line with early results [10].	10%
K7	Anomaly detection recall and sensitivity	>0.85	Recent methods for network anomaly detection achieve a precision-recall Area Under Curve (AUC) in the 0.66-0.88 range [11] leaving substantial room for improvement towards B5G systems. Having access to NI coordination, as well as to novel tools for a tailored design of AI, the NI-assisted anomaly detection mechanisms designed by DAEMON will target a 0.9 precision-recall AUC with at least 85% scoring in both precision and recall in scenarios of interest.	50%
K8	Vertical service response time	O(sec)	By taking advantage of in-network support, backhaul-assisted computing as a service for third parties shall contribute to reducing the response time of vertical services from minutes in current production systems to seconds with DAEMON NI-assisted functionalities.	30%
K9	Optimality gap of network management decisions	1%	Building on previous experience of the partners in anticipatory networking over long time horizons [12], DAEMON will ensure that decisions on network resource and function allocation occurring at periodicities of hours will perform very close (99%) to optimum oracles in scenarios of interest where the optimum can be defined. This will ensure that such decisions are precise enough to assist constructively faster NI, which use such longer timescale decisions (e.g. policies) as input.	35%

2.2 Evaluations

In order to assess the performance of the NI-assisted functionalities and demonstrate how they can achieve the KPI targets set out above, we are performing seven dedicated evaluations, as follows.

- ***Evaluation 1 (E1)*** demonstrates real-time control and non-real time orchestration of virtualized RAN (vRAN) services and resources. Experiments will have been carried out at sites T1 and T5 and with emulator S2, and focus on evaluating mechanisms that maximize the sustainability of dense vRAN deployments, minimizing their footprint and their operational and capital cost typically associated with greenfield deployments. *Associated functionalities:* Compute-aware radio scheduling.
- ***Evaluation 2 (E2)*** implements and demonstrates NI solutions that support network slice management and orchestration operations. Experiments have been conducted at site T8 and target the evaluation of the scalability, elasticity and stability of the NI-assisted automation and orchestration approaches developed by the project. *Associated functionalities:* Energy-aware VNF orchestration.
- ***Evaluation 3 (E3)*** validates tailored NI solutions for anomaly detection both in controlled environments and in a production core network. Experiments have been carried out at site T8 using ground-truth information on resolved incidences in the real-world network infrastructure of Telefonica, a major European operator. The NI-assisted anomaly detection designed within the project is integrated in a big data platform where the Mobile Network Operator (MNO) operations teams inject live telemetry data, allowing for real-time evaluations in realistic scenarios. *Associated functionalities:* Automated anomaly response.

- ***Evaluation 4 (E4)*** implements and deploy the NI-assisted solutions for service orchestration and resource allocation algorithms in the Edge micro-domain. Experiments shall leverage site T4, simulators S1 and S4 and datasets D1, D5, D6 and D9, validating the capabilities of the solutions developed by the project to dynamically orchestrate, allocate and deploy radio and network services. This setup is the ideal environment where to validate NI-driven tasks like, e.g., real-time radio technology classification or traffic classification, in a real-world scenario characterized by high-dimensional and very dynamic input data. *Associated functionalities:* Multi-timescale Edge resource management.
- ***Evaluation 5 (E5)*** tests NI-assisted solutions for anomaly response in very large-scale settings. These experiments take full advantage of datasets D7, D8, D11, D12, D13 and D14 in order to assess the capability of the NI to (i) trigger alarms in the presence of network anomalies within the available data, (ii) detect the root cause of such anomalies, and (iii) recommend network healing actions that include anticipatory resource and VNF reallocations based on capacity forecasting. *Associated functionalities:* Automated anomaly response, Capacity forecasting, In-backhaul support for service intelligence.
- ***Evaluation 6 (E6)*** validates the NI solutions designed for long-timescale operations, i.e., MANO, VNF orchestration and the associated allocation of resources. Experiments have built on dataset D1, since the performance of such network functionalities is best evaluated in large-scale scenarios. To this end, network load time series and other radio-cell-level and core network KPIs are used to demonstrate NI-assisted network function and capacity orchestration in a nationwide scenario. *Associated functionalities:* Self-learning MANO, Energy-aware VNF placement, Capacity forecasting.
- **Evaluation 7 (E7)** targets the demonstration of NI to configure Reconfigurable Intelligent Surfaces (RISs) in a controlled environment. Experiments will leverage site T9, where one transmitter will send a flow of data to one receiver in non-line-of-sight. A large codebook, optimized through NI, will be then tested to assess the passive beamforming gains of the reflective RISs. *Associated functionalities*: RIS control.

We refer the reader to Table 1 for a complete view of the association between KPIs and evaluations, as well as between evaluations and the tools they employ and the activities they entail. We clarify that, with respect to the Description of Action (DoA), we decided to add an extra evaluation for RIS control, namely E7. Our choice is motivated by the fact that during the implementation of RIS experimental platforms, we found such technology to aim more at performance improvement than sustainability. Also, we moved K6 from E1 to E7 accordingly.

3 Technical tools

In this section, ***we present the technical tools that are employed for the performance evaluation of the NI-assisted functionalities developed in the project***. We tell apart three categories, i.e., (i) simulations and emulators, (ii) experimental testbeds, and (iii) datasets. Tools in each category are detailed in separate subsections in the remained of the section.

3.1 Experimental testbed sites

The project relies on 12 relevant platforms for the experimental evaluation of the proposed solutions. Table 3 provides a list of these platforms, whose details are then expounded in the rest of the section. The table also indicates what evaluations and KPIs rely on each testbed.

Table 3. *Experimental sites available in the project, with related evaluations and KPIs.*

ID	Name	Short description	Related evaluation	Related KPIs
T1	Virtualized radio stack	This testbed will be used to demonstrate compute-aware radio scheduling solutions studied in T3.1	E1	K1, K2, K4, K5
T2	5Tonic	Large-scale facility for the testing of orchestration solutions using a commercial access network	E2	K1, K4, K5, K8, K9
T3	Multi-site 5G radio testbed	Multi-site 5G Testbed, which spans across two different sites in Barcelona and Madrid (Spain), offers a novel and unique framework for testing diverse Multi-access Edge Computing (MEC) applications	E3	K4, K5, K7, K8
T4	Smart highway	A real-life testbed for experimentation with vehicular communications and distributed edge computing	E4	K2, K5, K8
T5	SDR testbed with power meter	Software Defined Radio (SDR) testbed with a power meter to evaluate power consumption in virtualized RANs	E4	K1, K4, K9
T6	Dockerized srsRAN + Open5GS	Fully virtualizable solution for the creation of a 4G/5G network in a box, using srsRAN	E1	K1, K2, K4, K5
T7	VNF deployment and Edge Infrastructure	Testbed comprised of Bullsequana Edge nodes, which are for mobile computation, and high-performance commuters (Mellanox SN2100)	E1, E4	K1, K2, K3, K4
T8	Virtualized platform OSM and open stack	A set of servers on which an Open-Source MANO (OSM) and Openstack deployment is realized	E2, E3	K3, K7
T9	Reconfigurable Intelligent Surfaces	A set of custom-made RIS prototypes designed and built within DAEMON to demonstrate RIS control solutions	E7	K6
T10	Eclipse Zenoh testbed	The machines within this testbed will be used to test Zenoh's scalability, reliability and performance under different scenarios 10GbE are connected using a ring topology.	E4	K2
T11	Network capabilities, cloud resource testbed	Openstack-based multi-cloud infrastructure, consisting of 720+ CPU cores, 1700GB+ RAM and 120+ TB storage space, interconnected mostly via 10Gbps fiber/copper links	E6	K2, K5, K9
T12	P4 programmable tesbed for in-backhaul NI	Three Intel Tofino programmable switches deployed in a full 100-Gbps testbed with two dedicated servers for network emulation	E5	K3

3.1.1 Virtualised radio stack (T1)

Most of the results related to the evaluation E1 are performed on cloud-based deployment consisting of vRAN network functions. This deployment relies on srsRAN (formerly srsLTE), an open source software covering the lower protocols of the mobile network stack. More specifically, srsRAN provides the functionality of the Physical (PHY) up to Radio Resource Control (RRC) layer for evolved Node-B (eNB) or next Generation Node-B (gNB), while also supporting Fourth Generation (4G) or Fifth Generation (5G) User Equipment (UE). It is written in C/C++, and its configuration parameters cover a wide range of the base station and UE possible configurations. This software implementation runs, in a virtualized way, into

several servers, which have been used in the different evaluation tasks. At the time of writing, the virtualization infrastructure is composed by:

- 2 Supermicro SYS-E200-8D with Intel XEON processors and 16 GB of RAM each
- 1 Supermicro Superserver 6029U-TRT with:
    - 2 x Intel Xeon Gold 6226R, 2.9GHz, 16 cores/32 threads
    - 1x NVIDIA Tesla V100 32GB
    - 1x Intel Field Programmable Gateway Array (FPGA) PAC N3000 Vista Creek.

srsRAN uses Radio Frequency (RFR) frontends based on Software Defined Radio to provide connectivity between Ues and the base stations. srsRAN contributors have developed drivers for various commercial hardware RFR-frontends like URSP, Soapy SDR and BladeRF. For this testbed, we have available a set of 10 USRP B210, which can be arranged to conform different topologies and provide variable load to the infrastructure.

Additionally, some of the servers have hardware accelerators such as Graphical Processing Unit (GPU) or FPGA, to study the effect on the computing infrastructure of these hardware elements. Through this deployment we can measure, by probing the software implementation and the hardware infrastructure K1, K2, K4, K5.

3.1.2 5Tonic (T2)

5Tonic is a laboratory for enhanced 5G experimentation. The 5TONIC site is currently located at IMDEA Network premises in Leganés. This site provides a complete 4G and 5G network infrastructure, including 5G Non-Standalone (NSA) and Standalone (SA) support, in two different coverage areas. It is connected with other European sites (in the context of different ICT-17 projects such as 5G-EVE or 5G VINNI) as well as with Telefónica Spain Labs in Alcobendas, Madrid, and Telefónica I+D labs in Almagro Central Office, also in Madrid.

The Network Function Virtualization (NFV) infrastructure in 5Tonic is operated through an Open Source MANO orchestrator. Through 5Tonic, the Evaluation E2 can be performed, measuring the related KPIs K1, K4, K5, K8, K9 accordingly, achieving hence a large-scale evaluation. IMDEA, TID, and UC3M are members of the lab and can thus arrange evaluation activities leveraging the available infrastructure. More details are also available [13].

3.1.3 Multi-site 5G radio testbed (T3)

The multi-site 5G TID Testbed, which spans across two different sites in Barcelona and Madrid (Spain), offers a novel and unique framework for testing diverse MEC applications. In particular, its main goal building an automation framework for testing diverse edge solutions, where edge encompasses the portion of the mobile between the eNB and the CORD (Central Office Rearchitected as Data Center). This is attained by means of integrating generic purpose server pools where controllers and VNFs such as virtual core networks, virtual Baseband Unit (vBBU) are hosted in the form of a Virtual Machine (VM) or a container with open source a proprietary RAN software and hardware equipment. To provide the management and automation of the equipment, the Telefonica edge testbed leverages a set of open-source software tools for fast prototyping, automation, and testing. As for the emulation of the CORD, TID Testbed relies on four NFV servers physically located in Telefonica Datacenter premises, which form the physical infrastructure for testing the diverse MEC/NFV applications under evaluation. The orchestration of these NFV servers to host VNFs is based on Kubernetes container orchestration. A series of reconfigurable multi-purpose network server elements (2 servers based on Intel Xeon ES-2697 2.6Ghz with 56 CPUs and 8 Gigabit Ethernet cards and 2 servers based on Intel Xeon ES-2680 2.5Ghz with 48 CPUs and 4 Gigabit Ethernet cards.

As for the RAN segment, the testbed also offers different potential units as gNBs/eNBs, based on SRS and proprietary license-based (e.g., Amarisoft eNB and other eNB vendors). These 5G and 4G access networks can leverage the aforementioned NFV server.

As for UE, the testbed includes as well as a proprietary UE simulator (based on Amarisoft software) that allows the emulation of up to 128 Ues at the signal generation level, and real Ues based on the commercial 4G and 5G Phones. In terms of core network functionalities, our facilities offer different Evolved Packet Core (EPC) flavours, including open-source core flavours (e.g., open5GS) and proprietary flavours (e.g., Affirmed Networks).

3.1.4 Smart highway (T4)

We are developing a Proof-of-concept (PoC) real-life testbed environments modelling an AI-enhanced edge orchestration system. Figure 2, illustrates the PoC, which shall be employed for conducting realistic experimentation with automated and intelligent edge orchestration of Vehicular-to-Everything (V2X) services. The PoC leverages an existing Smart Highway testbed built along the E313 highway (Antwerp, Belgium) [61]. To create an edge network, we provide the Network Function Virtualization Infrastructure

(NFVI) by virtualizing computational resources in Road-side Units (RSUs) (RSU 3 and RSU 5), with the help of Kubernetes. These computational resources are used for deploying V2X services, and for performing their lifecycle management. The edge orchestrator is realized as an enhanced version of a Kubernetes master, because i) it supports cross domain operations, i.e., edge-cloud and edge-edge interaction, and ii) it is capable of training and using Machine Learning (ML) models for making intelligent decisions in an automated way. In our PoC, both edge orchestrator and NFVI can be deployed on the bare metal, as well as in Linux containers and virtual machines, which is a suitable practice for a shared experimentation environment such as testbed.

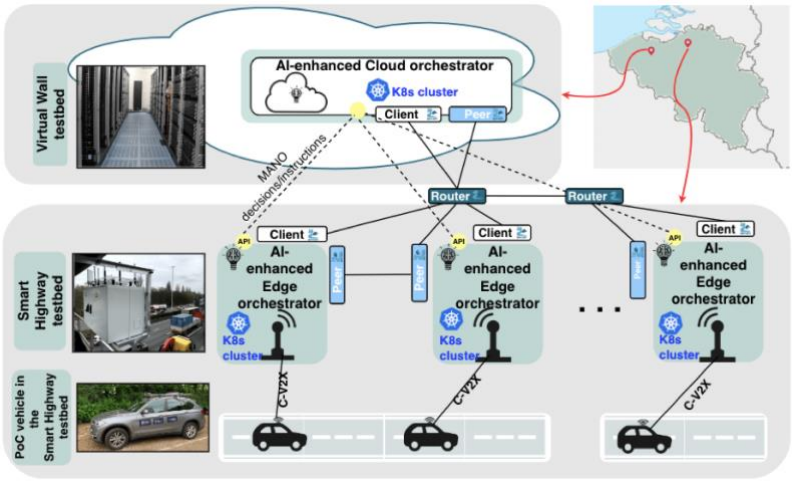


Figure 2. *PoC combining distributed testbed environments.*

The Cloud orchestrator is running on the bare metal on top of the Virtual Wall testbed, located in Ghent, Belgium (Figure 2). It is deployed as a web server (using Flask framework in python), which is capable of i) processing decision-offloading requests coming from the edge orchestrators, ii) location data processing and publishing on Zenoh, iii) injecting decisions on the north-bound interface of edge orchestrators to instruct them to proactively migrate/relocate services from one edge to another, and iv) receiving notifications from NIFs deployed on the cloud, which enhance their operations and help them make efficient decisions on managing underlying resources and edge orchestrators.

As different types of data need to be collected to feed ML models (e.g., computational and network resource utilization, energy consumption, KPIs measured at users' side, and users' locations), in this PoC we deploy MEC value-added services, as per definition in ETSI MEC, which perform data retrieval and pre-processing before publishing them on Zenoh. Given its minimal network overhead (as little as 5 Bytes), and its small footprint (around 60 kBytes on Arduino board), Zenoh is adopted in our PoC as a framework for data engineering pipeline. In particular, Zenoh provides a minimal set of primitives to deal with data in motion (e.g., real-time stream of vehicles' location/speed/destination), data at rest (e.g., historic data for vehicles' and edge nodes' computational resource utilization and energy consumption) and remote computations (e.g., on-demand calculation of the best route and speed limit). Each edge and cloud orchestrator acts as a subscriber for various types of data that can be stored on edges, and used for training or online learning/optimization.

Furthermore, concerning the vehicle as a client, our current includes one vehicle that is capable of communicating with the edge services via long range 4G (to be extended with 5G in the future). Thus, the client application is installed in the On-board Unit (OBU) of the vehicle, and it utilizes the Uu interface between User Equipment (UE) and gNodeB to exchange Cooperative Intelligent Transportation System (C-ITS) messages with services, and inform them about its location, speed, heading, and destination. The testing service that we deploy on the edges for the purpose of testing and demonstrating the work of PoC is the back-situation awareness V2X service, which addresses emergency situations on the road, thereby proactively informing vehicles on the road about the arrival of an ambulance. This service is containerized and designed in a cloud-native way, and thus orchestrated by the edge orchestrator.

3.1.5 Software-Defined Radio (SDR) testbed with power meter (T5)

This is a small testbed dedicated to collecting power consumption measurements on radio processing software. More specifically, the testbed is comprised of a 3GPP R10-compliant Long-Term Evolution (LTE) Base Station (BS), a UE, and a GPU server. The testbed is depicted in Figure 3.

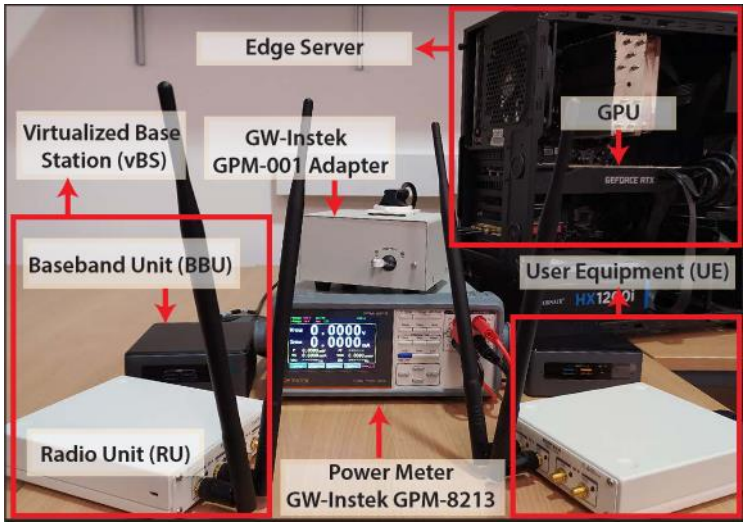


Figure 3. *SDR testbed with power meter.*

The BS and UE include an NI USRP B210 as Radio Unit (RU) and a general-purpose computer (Intel NUCs with CPU i7-8559U@2.70GHz) deploying the near real time RAN Intelligent Controller (RIC) (for the BS) and the baseband unit (BBU), implemented with the srsRAN suite (which emulates an O-eNB for experimentation). The virtualized Base Station (vBS) and UE are connected through SubMiniature version A (SMA) cables with 20 dB attenuators, and we adjust the transmission gain of the RU's RFR chains to attain different uplink Signal-to-Noise Ratio (SNR) values. The edge server is equipped with a CPU Intel i7-8700K @ 3.70GHz and a GPU Nvidia GeForce RTX 2080 Ti. The vBS and server are connected using a switch with Gigabit Ethernet technology.

To measure the power consumption of the BBU and the server, we use the digital power meter GW-Instek GPM-8213 with the GW-Instek Measuring adapter GPM-001. The server supports AI services. As an example, we have deployed Detectron2, developed by Facebook, which performs object recognition. Specifically, Detectron2 is configured with a Faster Region-based Convolutional Neural Network (R-CNN) comprising a ResNet backbone with conv4 layers and a conv5 head with a total of 101 layers. The UE sends to server images from the COCO data set [14] through the LTE uplink. The images are resized at the user side using the OpenCV library in Python. The bounding boxes and object classes are computed by Detectron2 and sent back to the Ues (LTE downlink).

We introduced two key srsRAN modifications. First, we modified the radio Medium Access Control (MAC) scheduler to implement different radio policies. Secondly, we integrated the O-RAN E2 interface to enforce such radio control policies on-the-fly and send consumed power consumption samples to the corresponding xApp. For the latter, we have added code into srsRAN to collect this information from the power meter. We have also implemented a PoC Near Real-Time (Near-RT) and Non Real-Time (Non-RT) RIC. We also have an interface to configure the GPU speed on-the-fly by using the Nvidia driver that allows us to set the maximum power management limit, ranging between 100 and 280W. This runtime configuration does not affect the GPU operation. Note that the actual GPU consumed power depends on its duty cycle.

3.1.6 Cloud-native mobile network emulators (T6)

As discussed in Section 3.1.1, srsRAN works with real radio frontends based on SDRs. Additionally, srsRAN have developed a software RFR-frontend based on ZeroM, an open source message queueing library written in C. When using this driver, the transmitted I/Q baseband symbols between UE and base station are transferred over various transport methods, like Inter-Process Communication (IPC) or Transmission Control Protocol (TCP) sockets. Choosing this driver avoids the need for high expertise in RFR channel configuration and facilitates the introduction of researchers who want to simulate a radio access network environment, but whose RFR channel is not their main area of interest or would be reluctant to invest in actual hardware transceivers.

Open5Gs, instead, is a very popular open source implementation of a mobile network core. Written in C, it stands as a reference among researchers and mobile telecommunications practitioners for experimentation and future enhancements. Currently supporting up to 3GPP 5G Release 16, it contains the most important components of the 5G Core and 4G EPC with Control-User Plane Separation (CUPS), meaning it can operate on both 5G NSA and SA modes, as it can serve both 4G eNBs and 5G gNBs. Its straight-forward build procedure makes its deployment in small-scale private networks very easy, while its modular architecture attracts its adoption into microservices-based cloud-native environment, that fits well with solutions such as Kubernetes.

3.1.7 VNF deployment and Edge Infrastructure (T7)

From the University of Málaga, we have at disposal the infrastructure of the I software institute that includes equipment for VNF deployment and edge infrastructure. Concerning the VNF deployment equipment, we have at disposal a dell Server with one 338-BSDH Intel Xeon Silver 4210, 2.2 GHz, ten cores, 20 subprocesses, 9.6 GT/s, a cache of 13.75 MB, Turbo, HT (85 W) DDR4 2400 MHz, two servers with 338-BTWN Intel Xeon Gold 5220S, 2.7 GHz, 18 cores/36 subprocesses, 10.4 GT/s, a cache of 24.75 MB, Turbo, HT (125 W), DDR4-2666. This equipment works with a 5G radio system with a Nokia AirScale System Module Indoor base band Unit, a Nokia Micro RRH 474147A and a Nokia Micro RRH 5GC001274. The edge infrastructure comprises seven nodes for Fog infrastructure with two CPU G62230R with 26 cores, 512 GB RAM and 2 GPUS Tesla V100s and 2 Bullsequana Edge Nodes from ATOS that are portable. The Bullsequana Edge nodes are for mobile computation and all the components are connected using high-performance commuters (Mellanox SN2100).

3.1.8 Virtualized platform, OSM and open stack (T8)

The testbed includes 3 servers and a total of 12 mini PCs as illustrated in Figure 4. In the main server, the OSM is deployed, while a set of capabilities including: a) AI enhanced MANO, b) Anomaly detection; c) Root Cause Analysis (RCA) and d) Performance diagnosis are also deployed as docker containers. Opendstack is deployed on top of the three other servers and the mini PCs. In detail, one server acts as the OpenStack Control, while the rest of the servers act as Openstack Compute nodes. The mini PCs have also Openstack capabilities. In the testbed a set of Services can be deployed in the GPU server which has the higher specifications: 64vCPUs, 128GB RAM, 2TB SSD.

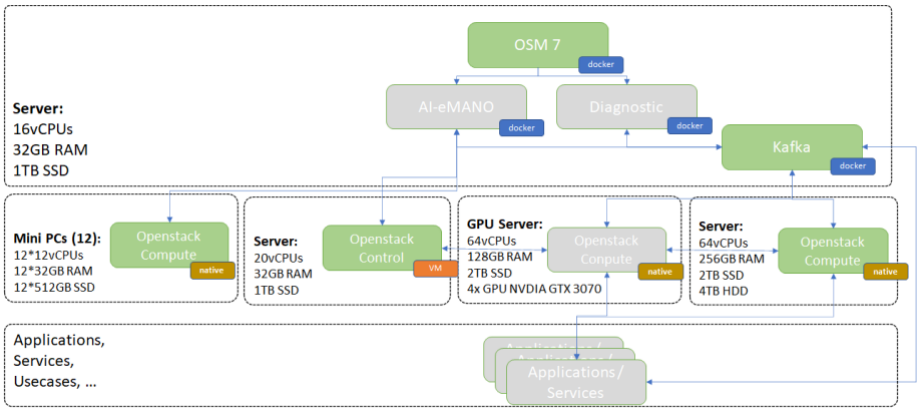


Figure 4. *WINGS testbed with OSM and open stack.*

In the testbed, NI functionalities can be developed and evaluated under different vertical scenarios, also the testbed allows collecting various metrics that include metrics from OSM/Openstack, the network, the application, and the functionalities for diagnostic, RCA and anomaly detection.

3.1.9 Reconfigurable Intelligent Surfaces (T9)

We have initiated the design of a RIS based on delay lines and RFR switches. The main purpose of our RIS design is to realize passive beamforming [15]: to reflect incoming waves impinging onto the surface with an arbitrary angle of departure defined by a controller with the goal of (i) (re-)focusing energy into the desired direction, and (ii) in the least energy-consuming manner. Consequently, no signal processing nor power amplification are permitted, and low-power electronic components must be carefully selected.

The basic element is a board made of a grid of *cell units* distributed in a 2D array with the ability to enforce phase shifts over impinging signals programmatically. By configuring an appropriate phase shift on each cell unit, we can attain beamforming gains passively, without resorting to power amplifiers or signal processors. This is shown in Figure 5.

Phase shifts are configured by a Micro-Controller Unit (MCU). The MCU is the only active electronic component in our design; hence, it is important to select a low-consuming microcontroller that is friendly to energy harvesting or other low-power sources. The MCU communicates with an external controller with a standard UART interface, a simple and low-power serial protocol. The MCU is not connected directly to each unit cell, which would not be feasible boards with a large number of unit cells. A more scalable approach is to connect each cell unit in the same row and column with a pair of buses, which we call "row/column selection bus", which *select* the cell unit to be configured. Then, another bus, the "phase configuration bus", communicates the desired configuration index (phase shift) out of a set of possible configurations for that selected unit cell. In this way, in an $N_x \times N_y$ board, we reduce the complexity of the design from $N_y \times N_x$ to $N_y + N_x$ connections.

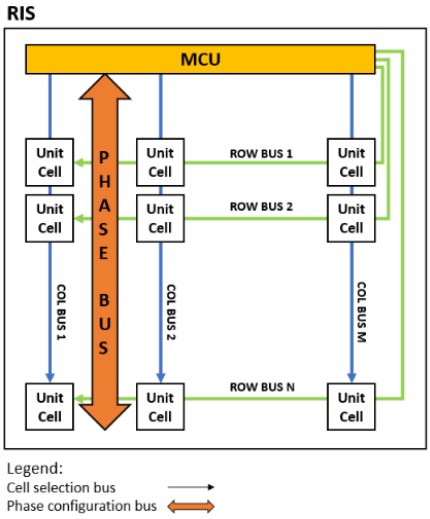


Figure 5. *RIS general overview.*

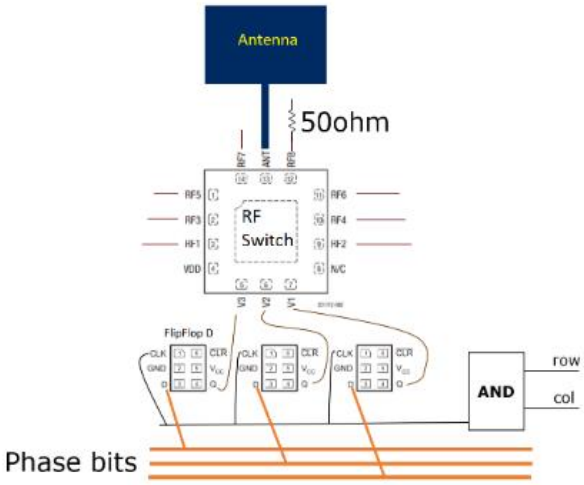


Figure 6. *RIS unit cell.*

As depicted in Figure 6, each cell unit is connected to both column/row selection buses through an AND gate. Hence, when the MCU sets a high voltage state in row x and column y, the MCU activates the configuration bus for cell unit (x,y) whereas all the remaining gates will output a low voltage state (0V), which de-selects them. A second relevant component in the design of our cell unit is a flip-flop D, which has the ability to store 1 bit as long as it is powered. When a flip-flop senses a rising edge, it updates the value in memory and then sends it out as output. To this end, the high state exiting the AND gate works as a rising edge for the flip-flop.

We designed our RIS with a 3-bit resolution in the phase shift configuration space. Therefore, each cell unit integrates three flip-flops, and we use three 1-bit phase configuration buses as shown in Figure 6. The third important component in each unit is an RFR switch, which can redirect the RFR signal received in an input port towards one output port selected by the configuration ports. This is also shown in Figure 6. Each configuration port is directly connected to one configuration bus, as shown in the figure. Moreover, each output port is connected to an open-ended transmission line, each with a (different) length calculated to provide a specific time delay on the bouncing signal, and hence provide a desired phase shift. We reserve one configuration output to connect a resistor matching the characteristic impedance of line and the switch, which dissipates the incoming signal and prevents the signal to be irradiated back. We call this configuration "absorption state" and enables us to change the size of the surface area that can reflect signals and hence lets us virtually change the size of the RIS, which is useful for a number of use cases. The last component of the unit is a patch antenna, a particularly cheap antenna with low gain that is the ultimate responsible of interacting with electromagnetic waves.

Our approach is modular: multiple boards can be connected through a common UART bus, and each of them can be singularly addressed by the external controller using different identifiers. The disposition of the unit cells across cells within and across boards have been carefully designed to have a separation of $\lambda/2$, where λ is the wavelength of the operating frequency. This provides us an ideal approach to increase/decrease the physical area of our structure without compromise inter-antenna distance.

The next steps of the testbed development will be (i) designing and printing each component described above in PCB (Printed Circuit Board), and (ii) empirically characterizing the resulting device. This has several advantages such as low cost, fast production time, and suitability for large-scale implementation.

3.1.10 Eclipse Zenoh testbed (T10)

The Eclipse Zenoh testbed is composed by four servers, interconnected via 100 GbE fiber links as illustrated in Figure 7. In each server a variable number of Zenoh routers, peers and Zenoh Flow runtimes is deployed based on the different experiment.

In the testbed, Zenoh's scalability, reliability and performance metrics are evaluated under different scenarios, such as different topologies and payloads, leveraging on virtualization of both computing and networking fabric. The same testbed is also used to evaluate Zenoh-Flow capability to run NI algorithms and to leverage on heterogeneous computing devices such as CPUs and GPUs.

Different metrics are collected in the testbed including: (i) Zenoh latency and throughput metrics; (ii) Zenoh scalability metrics; (iii) Zenoh footprint metrics; (iv) Zenoh Flow latency metrics; and, (v) Zenoh Flow footprint metrics.

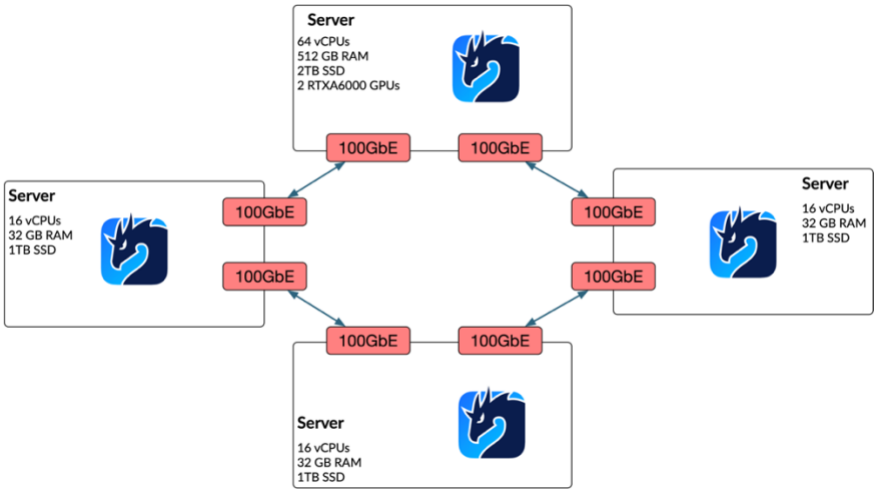


Figure 7. *ADLINK's testbed with Eclipse Zenoh installed.*

3.1.11 Network capabilities and cloud resources testbed (T11)

OTE will develop in the project a cloud testbed for hosting of a subset of the VNFs, which will be developed in DAEMON project. The testbed includes an Openstack-based multi-cloud infrastructure. In the current setup Openstack Queens 23oolean is available on Ubuntu Server 16.04/18.04 LTS. The testbed collectively consists of >720 CPU cores, >1700GB RAM and >120TB storage space, and is interconnected (mostly) via 10Gbps fiber/copper links. Compute and storage resources can be made available for hosting relevant services. The setup can be split into one or more cloud slices (controllers/ compute nodes/ hypervisors) of various sizes, either in bare metal or virtualized form, in order to allow high degrees of freedom for customized configurations to meet projects' needs. Moreover, a set of small cells are also available. The next figure presents a conceptual view of OTE's topology for the testbed which will be used in the DAEMON project. OTE will also provide the necessary networking capabilities (e.g., VPN access, certificates, etc.) for access provision to the involved partners. Figure 8 summarizes the testbed.

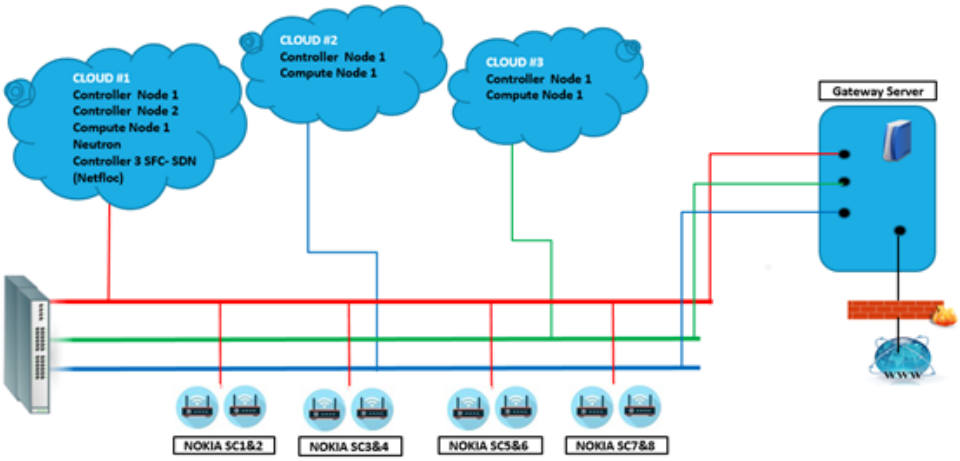


Figure 8. *Topology of OTE's cloud testbed.*

3.1.12 P4 programmable testbed for in-backhaul NI (T12)

As part of the DAEMON activities, we developed a cutting-edge testbed to perform experimental research on in-backhaul NI. The main purpose of the testbed is to evaluate the performance of machine learning algorithms that run at line-rate in the user plane by means of implementation on real programmable switches. The first use case that we already started to work on (but not the only one that we planned) is the detection of malicious traffic to provide very fast response to anomalies. For such use case, the target KPI is K3 and the evaluation that we perform is E5.

The testbed is composed of two servers and three P4 programmable switches equipped with Tofino application-specific integrated circuits (ASIC). The hardware is installed on a rack and connected to the Internet via a non-programmable Top of Rack (ToR) switch. In the current setup, the ToR switch is connected to both servers on a dedicated subnet that we use to access the servers remotely and to provide Internet connection. The three programmable switches are connected to both servers and to each other in an isolated and fully-connected subnet that is used only for experiments. We run virtual hosts, via either Docker containers or virtual machines, on both servers. Communicating to each other,

the virtual hosts generate traffic that passes through the switches and that we monitor and classify directly into the user plane. The control plane is implemented in one of the two servers, which is responsible of the switches' configuration and operation, of the injection of P4 compiled code and of the runtime control of the switches.

Figure 9. *Picture of the P4 programmable testbed hardware installed on the rack.*

The relevant hardware components of the testbed are shown in Figure 9 and are detailed as follows:

- 2x DELL PowerEdge R7515, 2RU equipped with: CPU AMD EPYC 7402p, 2.8 GHz, 24 cores, 128M cache; RAM RDIMM 128 GB, SSD 480 GB; 2x Mellanox ConnectX-5 dual port, QSFP28 (40/100 GB);
- 3x Edge-core Wedge 100BF-32QS, 1RU equipped with: Intel Tofino BFN-T10-032Q; Quad-pipe programmable packet processing pipeline for 6.4 Tbps total bandwidth; 32x ports QSFP28 (40/100 GB); CPU Intel x86 Xeon D-1548, 8 cores; SSD 2 TB.

With regards to software, we rely on open-source operating systems for all the hardware. We installed Ubuntu Server on both servers. Each switch is equipped with the full software stack to enable a full-fledged SDN platform: Open Networking Linux (ONL) as OS, and Stratum as a thin OS for remote configuration and control.

3.2 Simulators and emulators

The project counts with 4 simulation or emulation platforms, which are summarized in Table 4, and are fully detailed in the following. The table also indicates which evaluations and KPIs rely on each simulator.

Table 4. *Simulators and emulators available in the project, with related evaluations and KPIs.*

ID	Name	Short descritpion	Related evaluation	Related KPIs
S1	Edge/Cloud simulator	Used for resource management performance evaluation	E4	K4
S2	P4 programmable RAN	Include disaggregated RANs, P4 bmv2 switch and NBL CN	E1	K8
S3	System level simulator	Advanced component validation and optimization	E3	K7
S4	EnergyEdgeCloudSim	Extension of EdgeCloudSim environment that considers energy consumption	E4	K1, K2, K3, K4

3.2.1 Edge/Cloud simulator (S1)

This Edge/Cloud simulator, named DynamicSim, is based on Sim-Diasca (Simulation of Discrete Systems of All Scales) [16]. Sim-Diasca is a general-purpose, parallel, and distributed discrete-time simulation engine written in Erlang language. Sim-Diasca allows the simulation of complex systems focusing on scalability, in order to handle simulation cases that may be very large (potentially involving millions of interacting instances of models), while still preserving essential simulation properties, like causality, total reproducibility and some form of ergodicity. Figure 10 shows the Sim-Diasca modular architecture. Its internal modules are in charge of synchronizing time between the actors, evolving the system state, sending and receiving messages to and from the controller, and managing the results.

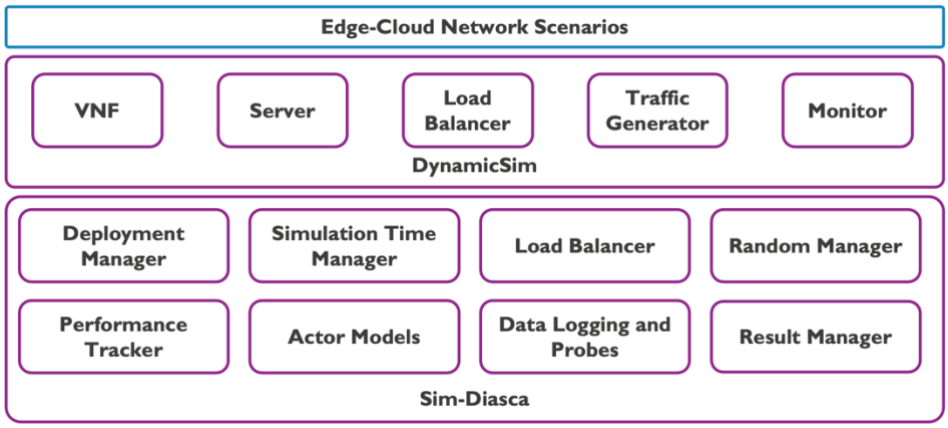


Figure 10. *Sim-Diasca Architecture.*

Sim-Diasca is based on the actor model; therefore, every single concept to be simulated is called an actor. Actors communicate with each other through messages. In response, actors can make decisions, create more actors, send messages to other actors, set how to respond to next messages. By using the actor model, different use-case simulations can be created. In a simulation case, the duration of a time step is user-defined. Within a time step, the actors simulate its functionality representing the work done in such a duration. After each actor finishes its simulated work, the time manager increases the time step by one, and the simulation goes to the next tick. At the beginning of the simulation, an initial set of actors are generated based on the defined simulation case.

In our case, we created a layer on top of Sim-Diasca, called DynamicSim, in which we define an actor model for Virtual Network Functions (VNFs), servers, load balancers, traffic generators and monitor modules. Specifically, traffic generators and monitor modules act as an interface between the actors in DynamicSim and high-level functions defined in other programming languages. Finally, several user-defined simulation cases can be designed in a higher layer. Thanks to Sim-Diasca generality, end-to-end metrics can be defined per use-case simulations. For example, using DynamicSim we can obtain low level metrics such as the number of active VNFs, the CPU consumption of each VNF, and the peak latency of the processed traffic and high-level metrics such as Service Level Objective (latency) violations.

3.2.2 P4 programmable RAN (S2)

The P4 programmable RAN is an emulation platform that can be used to execute common user applications (e.g., YouTube, web browsing, file transfer, etc.) over real protocol stacks and standardized procedures. Specifically, it is built on top of several key components to formulate an end-to-end network spanning the disaggregated radio access, transport, and core networks: (1) OpenAirInterface (OAI) [17], (2) P4-based switch [18], and (3) Nokia Bell Labs core network. Furthermore, to serve multiple Ues and execute UE-specific applications, we use the OAI-based UE modem interconnected with the OAI-based DU to demodulate/decode traffic and forward per-UE traffic to the corresponding virtual machines (VMs). Figure 11 shows all of the component: Server 1 hosts RAN, Transport Network (TN), and Core Network (CN) entities, Server 0 hosts the UE dashboard and message adaptor, and the remaining servers are used to host all Ues. Note that each UE is placed as an individual VM, and thus they are isolated from each other.

To provide more insight into the platform, the L2-sim mode is used between OAI-DU and OAI-UE, in which their MAC layers are connected directly using the nFAPI interface, and their physical layer processing is omitted for simplicity. Nevertheless, to emulate the physical layer behaviors, two additional schemes are added: (1) Time-varying channel quality model and (2) a transport block retransmission model. The former aims to provide configurable CQI patterns/distributions for each UE (e.g., fixed pattern, uniform random distribution, or Markov chain), whereas the latter applies the ARQ scheme (i.e., no redundancy version) to retransmit uncoded MAC SDUs using three parameters (i.e., first transmission acknowledgement probability, retransmission acknowledgment probability, and maximum retransmission count) following the standardized retransmission timing for all HARQ processes. Therefore, in our current setup, up to 32 Ues can be managed by the UE dashboard server and different user applications (cf. Section 4.1.3) in the app repository can be independently executed within each VM.

Finally, the P4-Programmable RAN emulation platform is used to evaluate the related KPIs for application-aware radio scheduling. The message adaptor at Server 0 can capture real-time user-plane information feedback from the system (at the granularity of each radio bearer) and provide the required dataset to facilitate the design of the corresponding algorithms.

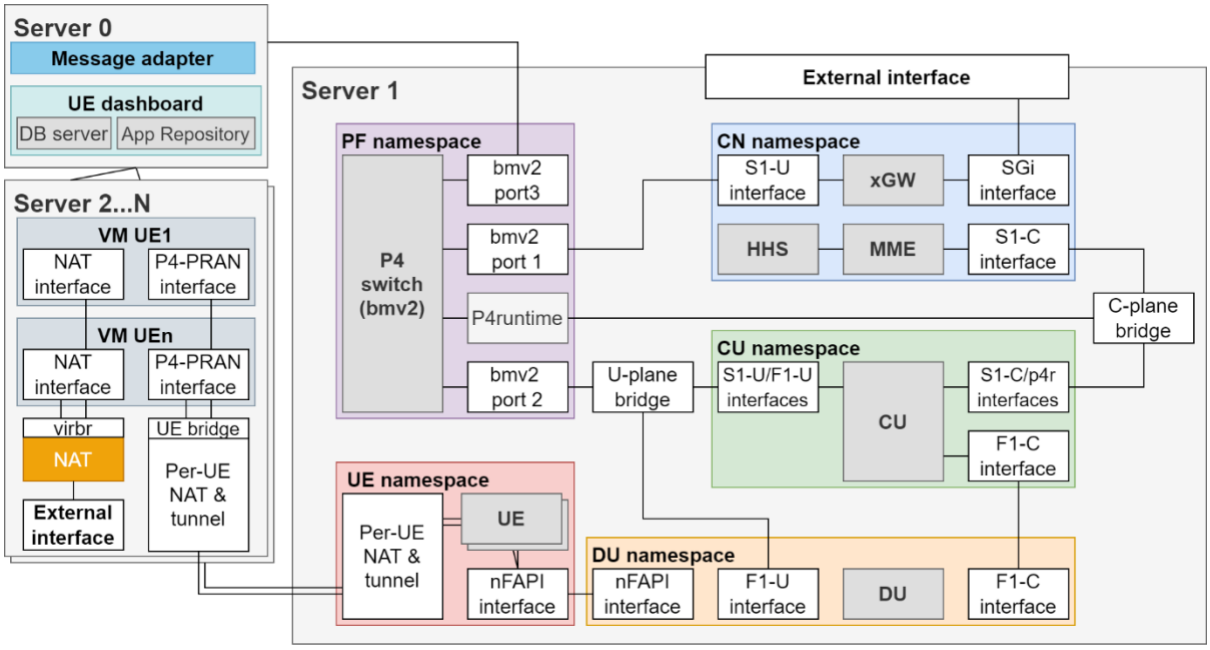


Figure 11. *P4-PRAN emulation platform.*

3.2.3 System-level simulator (S3)

The system-level simulation platform for 5G is a Discrete Event Simulation (DES) environment for the simulation of heterogeneous networks. Also, the platform is extended with new features to support the new functionalities of 5G. The main modules supported are macro cells, small cells and Ues nodes. Based on the DES approach, created events are the basic signaling events, mobility events, application layer events, and also system level events that enable the collection of measurements and the control of auxiliary artifacts (graphics, controls etc.). The tool has the potential of simulating various scenarios under different assumptions/ conditions. Through the flexibility of available modules, it is possible to customize various parameters. As such, the simulator involves a series of input parameters such as customizing the size of the simulation area; the area type (e.g., dense urban scenarios, etc.); the number and position of 3-sectorized macro base stations and their inter-site distances (ISDs); the number and position of small cells per macro base station; the number and position of end-user devices; the mobility of the end-user devices etc. that are used for various testing simulations and components.

The system-level simulation platform considers aspects related to configuration, environment models, network (simulated system) models, analytics, event management. All these are managed via a user-friendly graphical user interface (GUI), as depicted in Figure 12, and are presented next.

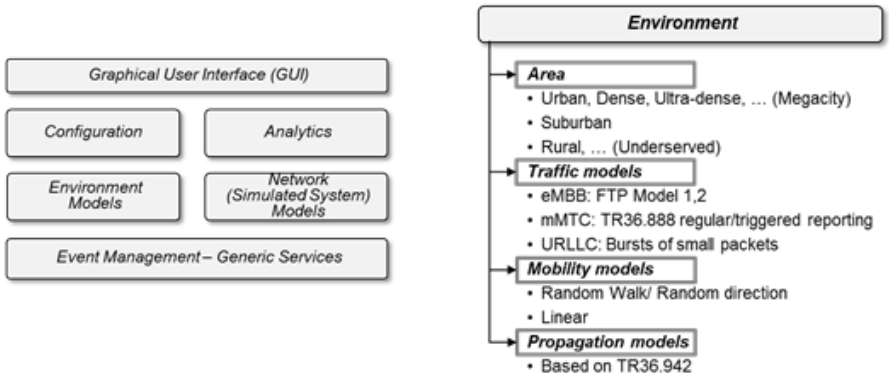


Figure 12. *System level simulator.*

Environment models and configuration: An important aspect of system-level simulations is to specify the simulated system, designate the environments and select analytics. Environment concerns aspects related to traffic (e.g. proper modeling of eMBB, mMTC etc., anticipated load, mobility and radio conditions (e.g. propagation models). This is triggered by the fact that project use cases deal with megacities and underserved areas and as a result, different traffic characteristics apply depending on the use case. Such aspects will be properly documented for the considered use cases in order to consider them in the simulations later on.

Network (Simulated System) models: System aspects include considerations relevant to network deployment (e.g. small cells and macro cells for use cases in underserved and megacities). Also, spectrum aspects are considered for utilization of bands below 6GHz and to be expanded in mm-wave as well. Abstraction of PHY/MAC is taken into account. Radio Resource management (RRM) algorithms are also considered.

Analytics: The simulation results will be evaluated against the KPI targets (e.g. in terms of throughput, latency). The results are analyzed and visualized.

Event Management: An event may be distinguished by time, location, type (e.g., session set up, call request, packet transmission), services, devices, users and supplementary info. Details on event management are provided later on in this paper.

Graphical User Interface (GUI): A user-friendly GUI is essential for easy handling of simulations and demonstrations. The GUI consists of intuitive tabs, text boxes and input fields in order to create an easy-to-use environment for data input as well as extraction of results by visualizing results in graphs and charts.

Overall, the simulator allows supporting ambitious use cases: for instance, use case families in NGMN that include broadband access in dense areas and everywhere (eMBB), massive Internet of Things and machine-type communications (mMTC) as well as ultra-reliable communications (URLLC). For the needed representation/ modeling of such aspects, environment models shall take into account area aspects, traffic, mobility and propagation models based on the classification. These features are also captured in Figure 12.

3.2.4 EnergyEdgeCloudSim (S4)

EnergyEdgeCloudSim is an extension of the tool EdgeCloudSim [19] for energy consumption measurements. The original tool, EdgeCloudSim, is a simulation environment specific to Edge Computing scenarios where it is possible to conduct experiments that consider both computational and networking resources. Our extension extends the nodes' information with parameters related to energy consumption.

EnergyEdgeCloudSim considers both dynamic and idle energy consumption. The dynamic energy consumption model distinguishes between computational and communication energy consumption. The expression to estimate the computational energy consumption includes CPU usage storage, and RAM, being the CPU usage the most influential factor [20]. The following is a list of the equations that support our energy consumption model (in Joules) for a task i which is running in a node n associated to computation ($eComp_{n,i}$), data sending and receiving ($eSend_{dataUpi,n}$ and $eRcpt_{dataDowni,n}$), and the energy consumption for idle or sleeping nodes ($eIdle_n$ and $eSleep_n$):

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

$$eIdle_n = \alpha_n eMax_n tew_n$$

$$eSleep_n = \beta_n eMax_n tew_n$$

$$eSend_{dataUpi,n} = P_n^{Tx} \frac{dataUp_i}{R_n^{Tx}} ew_n$$

$$eRcpt_{dataDowni,n} = P_n^{Rx} \frac{dataDown_i}{R_n^{Rx}} ew_n$$

Concerning the equation for $eComp_{n,i}$, $eMax_n$ is the energy consumption for a fully-utilized server in terms of CPU; v_i the CPU utilisation ratio (0-1) for the task i; w_i the CPU cycles required to compute task i; α_n is a value between 0 and 1 that represents the fraction of the idle energy consumption for the node n; and $eDeploy_n$ is the (fixed) amount of energy required by node n to create a container. The equations for $eIdle_n$ and $eSleep_n$ considers additional factors like the time t in seconds and β_n which is the fraction of the sleep energy consumption. Expressions to measure energy consumption for communication consider the transmission power (P_n^{Tx} and P_n^{Rx}) and the transmission rates (R_n^{Tx} and R_n^{Rx}). Finally, ew_n (energy weight), presented in all the expressions, allows to select the importance of saving energy in each device separately. Thus, if you intend to reduce energy consumption in battery-powered devices, simply set that variable to 1 for these devices and 0 for the rest of the nodes.

These models have supported the development of an approach for orchestration and auto-scaling that minimizes energy consumption (see Figure 13). The system serves the users' requests, who demand the functionality offered by a series of applications contained in a repository (e.g., DockerHub). An orchestrator manages the edge nodes (e.g., Kubernetes [1]) that automates the deployment, management, scaling, interconnection and availability of applications. The master nodes (there may be more than one) are responsible for orchestrating the workloads between the associated devices (worker

[1] https://kubernetes.io/es/docs/home/.

nodes)–master nodes can also be worker nodes simultaneously. Once demanded, the orchestrator assigns an application to an edge node, which executes it packaged in a container (e.g., Docker). The scheduler decides which worker node will run that container. We modify this scheduler to assign tasks to the most energy-efficient nodes to minimise energy consumption. Periodically, the master node (or one of them) requests the Essential Node Identifier module, which starts the proactive horizontal auto-scaling. This module receives the expected workload (number of requests) and the current state of the infrastructure, and, using this information, it determines the demand of nodes in the next time interval (defined by the infrastructure administrator). This module also has access to the applications' data contained in the repository. The auto-scaling process can run on a node in the infrastructure or an external node (even in the cloud). Once the master node receives the information on the nodes to be kept active, it is responsible for putting those not considered essential on sleep mode. In practice, nodes are put on sleep mode through SSH commands and wake up again using Wake on LAN/WLAN.

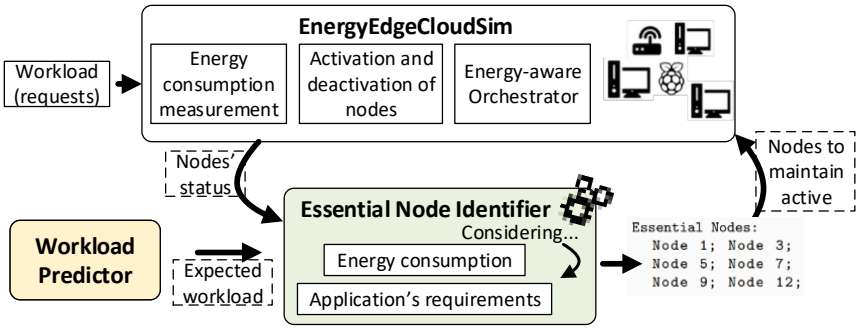


Figure 13. *Orchestration and auto-scaling in EnergyEdgeCloudSim.*

Currently, we are performing experiments to measure the amount of energy saved through our proposal comparing different orchestration policies. In addition, we are working on the issue of the number of failed requests as the reduction in the number of available nodes can lead to a lack of available resources. We are analysing the impact of more or less resource preservatives policies in this direction. Finally, we are studying the scalability of our approach concerning the problem size. With this goal, we are developing a benchmark version of the Essential Node Identifier module that can work with random VNF's requirements to increase the number of expected NFVs and the number of nodes.

In the context of DAEMON, EnergyEdgeCloudSim is part of our approach to validating NI for Edge Orchestration (E4). Our goal is to demonstrate improvement in the following KPIs: K1-VNF for energy consumption reduction, K2-saving of computational resources at the edge, K3-response time of AI-based NI algorithms and K4-operating expense saving. We plan to collect SM1, SM3, TM2, TM4 and TM5. We have already collected SM1 (throughput) and SM3 (energy consumption).

3.3 Datasets

The evaluations carried out in the project build upon 14 datasets to date. Table 5 provides a list of these datasets, whose details are then expounded in the rest of the section. The table also indicates what evaluations and KPIs are associateds to each dataset.

Table 5. *Measurement datasets available in the project, with related evaluations and KPIs.*

ID	Name	Dataset availability	Source	Data velocity	Data volume	Related evaluation	Related KPIs
D1	MNO radio performance	Private	Real	Depending on the specific data feed (e.g., hourly, per 15 min, real-time)	Order of TB per day	E5	K5, K7, K8, K9
D2	End-user performance	Private	Real	Depending on the type of data	Depending on the type of data	E6	K2, K4, K5, K6, K9
D3	Service-level traffic demand	Private	Real	1 sample per minute	Order of TB	E1	K2, K4, K9
D4	vRAN performance and power consumption	Open source (Link)	Real	1 sample every 20 seconds	12.2 MB	E4	K1, K2, K3

D5	Edge Dataset	Open source (Link)	Real	1 Sample per minute	2.8 MB	E4	K2
D6	Wireless interactions in multiple BSS using Channel Bonding	Open source (Link)	Synthetic	3 sample per minute	20 KB per deployment	E4	K5, K8
D7	Intrusion Detection Evaluation Dataset	Open source (Link)	Real	Variable[2]	8.3 GB	E5	K3
D8	IPX Signaling Dataset for IoT	Private	Real	Signaling dialogues arrive every 5min	Order of GB per day	E5	K5, K7, K8, K9
D9	YouTube file requests	Open source (Link)	Real	1 sample every 5 minutes	Order of MB per day	E4	K3, K4
D10	GEC case study	Open Source (link)	Real	1 sample every second	14,3 MB	E2	K1
D11	IoT devices dataset	Open source (link)	Real	Variable[2]	12.7 GB	E5	K3
D12	Applications and protocols dataset	Open source (link)	Real	Variable[2]	581 MB	E5	K3
D13	Malicious attacks dataset	Open source (link)	Real	Variable[2]	22.6 MB	E5	K3
D14	Malicious packets dataset	Open source (link)	Real	Variable[2]	10 GB	E5	K3

3.3.1 MNO radio performance (D1)

We collect dataset D1 from an operational mobile network in the UK. The cellular network we study supports 2G, 3G, 4G and 5G mobile communication technologies. In Figure 1, we illustrate a high-level schema of the MNO architecture. Such a network Can be simplified to consist of three main domains: (i) the Cellular device (in our case, the smartphone used as primary device by end-users), (ii) the RAN and (iii) the Core Network (CN).

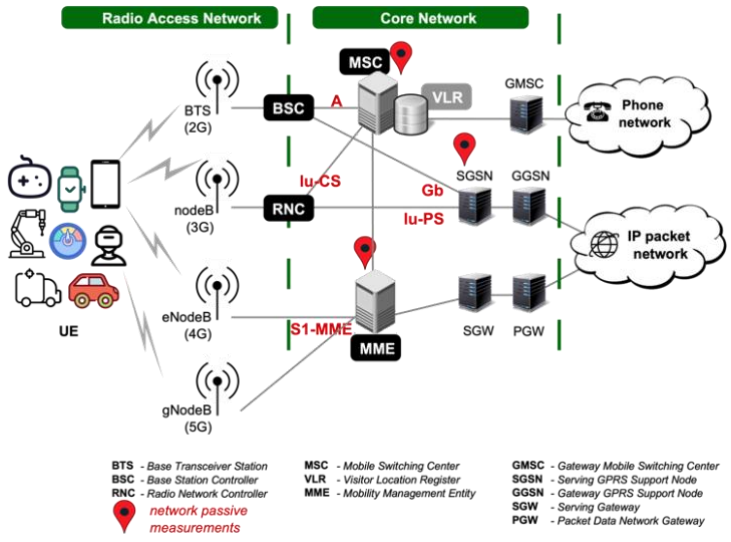


Figure 14. *High-level architecture of the measurement infrastructure integrated in the cellular network.*

[2] The dataset is composed of .pcap files with packet traces that have a high variability of time of arrival.

Our passive measurement approach relies on commercial solutions the MNO integrates within its infrastructure. The red pins in Figure 14 mark the network elements that we monitor, namely the Mobility Management Entity (MME), the Message Sequence Chart (MSC), the Serving GPRS Support Node (SGSN)/Serving Gateway (SGW), and the Cell Sites. We collect control plane information for both voice and data traffic from the total population of devices connected to the MNO's radio network, as well as KPIs of cell sites. From this measurement infrastructure, we capture various data feeds, described next. These feeds are aggregated at postcode level or larger granularity.

General Signaling Data. We capture the activity of the users in the control plane for the different Radio Access Technologies (RATs) supported By the cellular provider. The data includes control plane signalling messages related to events triggered by the MNO's subscribers, including Attach, Authentication, Session establishment, Dedicated bearer establishment and deletion, Tracking Area Update (TAU), ECM-IDLE mode transition, Service request, Handover and Detach. Each event we capture carries the anonymized user ID, Subscriber Identity Module (SIM) Mobile Country Code (MCC) and Mobile Network Code (MNC), Type Allocation Code (TAC) (the first 8. Digits of the device IMEI, which are statically allocated to device vendors), the radio sector handling the communication, timestamp, and event result (success or failure).

Devices Catalog. Using a commercial database provided by Global System for Mobile communications (GSM) Association (GSMA), we map the device TAC to a set of properties such as device manufacturer, brand and model name, operating system, radio bands supported, etc. With this information, we are able to distinguish between smartphones (likely used as primary devices by the mobile users) and Machine-to-Machine (M2M) devices. Radio Network Topology. To account for potential structural changes in the radio access network (e.g., new site deployments), we rely on a daily snapshot of the network topology. This includes metadata (location and configuration) and the status (active/inactive) of each cell tower.

Radio Network Performance. This dataset includes various hourly KPIs, including average cell throughout, average user throughput, average percentage of resources occupied, average number of users, total volume of data traffic uplink/downlink and total volume of conversational voice traffic.

Mobility Metrics. Using the signalling dataset described above, we can associate each (anonymized) user to a radio tower throughout the time they are connected to the MNO's network. Based on the radio network topology, we further attach to each radio tower its geographic location (postal code and approximate coordinates). With this, we then generate aggregated mobility statistics over six disjoint 4-hour bins of the day, for roughly 22 million native users aggregated at postcode level. We then compute two mobility metrics: entropy and radius of gyration. The combination of these metrics gives a wide view of changes in mobility: while entropy measures the repeatability of movements, radius of gyration is an indication of the distance travelled.

3.3.2 End-user performance (D2)

OTE is collecting large-scale datasets in order to provide input for the DAEMON automated anomaly response-related algorithms. The measurement data is aggregated over geographical areas (e.g., from antenna sector to urban statistical zones), temporal intervals, or flows generated by a significant number of users, depending on which aggregation preserves data utility for the subsequent analysis carried out by the project partners. The process of collection and aggregation occurs at the MNO premises. The DAEMON activities only concern data analysis, and the partners will only access the aggregated data, which does not contain personal information nor data with the potential to identify individuals. An indicative set of data to be collected includes: (1) Up/down link volume; (2) Up/down link duration; (3) Voice duration; (4) Number of connections; (5) Pedestrian / vehicular minutes of use; (6) Connection drops; (7) Handover-related information; (8) Traffic volumes (in bytes), aggregated per minute, for each antenna sector, ideally over multiple cities; (9) Traffic volumes disaggregated across service categories (e.g., video streaming, social media, etc.) or individual services (e.g., YouTube, Tik Tok, etc.). The aforementioned list is under consideration and will be revisited by the project partners.

3.3.3 Service-level traffic demand (D3)

This dataset was collected in the core network of a major European mobile operator, and made available under a Non-disclosure Agreement (NDA) to IMDEA. The dataset describes the mobile traffic generated by the whole subscriber base of the operator (which has a 34% average market share in the target country) over the whole territory of a European country. The data covers three months in 2019. The time frame of the data allows capturing a variety of mobile traffic dynamics.

A simplified representation of the operator 3G/4G mobile network architecture is portrayed in Figure 15. The figure is limited to the 3G UTRAN and packet switched core, and to the 4G EUTRAN and EPC, as our focus is on data traffic, and 5G was not yet operational in the target country by the time of the data collection. The data was recorded by passive probes at the Gn and S5/S8 interfaces of the Gateway GPRS Support Node (GGSN) and of the Packet Data Network Gateway (P-GW). The 3G and 4G gateways were conveniently co-located in the operator infrastructure, which eased the probe deployment, management and synchronization. The probes inspected IP traffic on the GPRS Tunneling

Protocol user plane (GTP-U), and extracted information on the transport- and application-layer protocols of each user session. The specific mobile service associated to each IP session was detected by the mobile network operator via Deep Packet Inspection (DPI) and multiple proprietary fingerprinting techniques, each tailored to a specific traffic type. These operations can classify 88% of the mobile traffic.

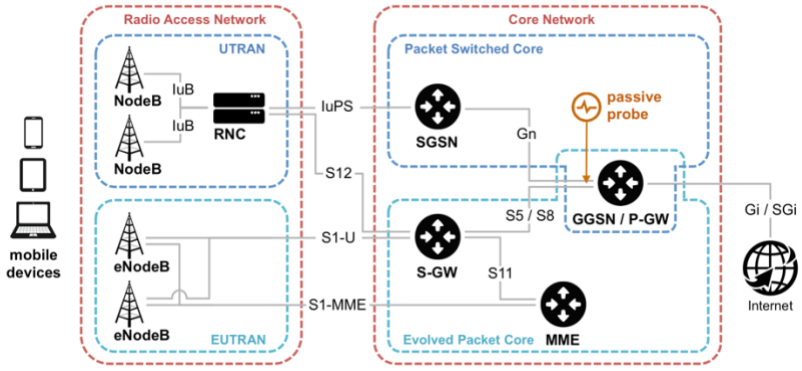


Figure 15. *Simplified 3G/4G mobile network architecture.*

Geo-referencing of the IP sessions, and of the corresponding mobile service usages, was performed by examining the User Location Information (ULI) contained in the 3G Packet Data Protocol (PDP) Contexts and 4G Evolved Packet System (EPS) Bearers. These data structures are transferred over the GPRS Tunneling Protocol control plane (GTP-C), which also transits through Gn and S5/S8 interfaces, making their inspection straightforward. User localization was further refined by mixing the data collected in the core network with signaling information gathered at the radio access, and pinpointing the NodeB or eNodeB each subscriber was associated to over time.

Ultimately, the dataset we use in the context of the DAEMON activities study consists of the daily volume of data traffic served by individual NodeB or eNodeB. The level of spatiotemporal aggregation ensures that no data subject can be re-identified, and that the data does not configure as personal data in the acceptation of the General Data Protection Regulation (GDPR) [21]. Therefore, the dataset and research do not involve risks for the mobile subscribers, while they provide new knowledge about the potential existence of a second-level digital divide in France, which may benefit a more informed social policing.

The raw traffic measurements used to derive the dataset above were stored and aggregated in a secure platform at the operator premises, in full compliance to article 89 of the GDPR, under the supervision of the Data Protection Officer (DPO) of the operator, and upon authorization by the French National Commission on Informatics and Liberty (CNIL). The raw measurements were deleted upon data aggregation.

3.3.4 vRAN performance and power consumption (D4)

This dataset provides a set of measurements of the performance and power consumption of a vBS using the srsRAN software in a local testbed (T5). The dataset was collected by configuring the vBS and UE in order to fix the conditions in the uplink and the downlink in terms of traffic load, channel quality, MCS, and airtime [22]. Then, we fix each configuration for one minute while the system takes measurements that later are processed to obtain its statistics.

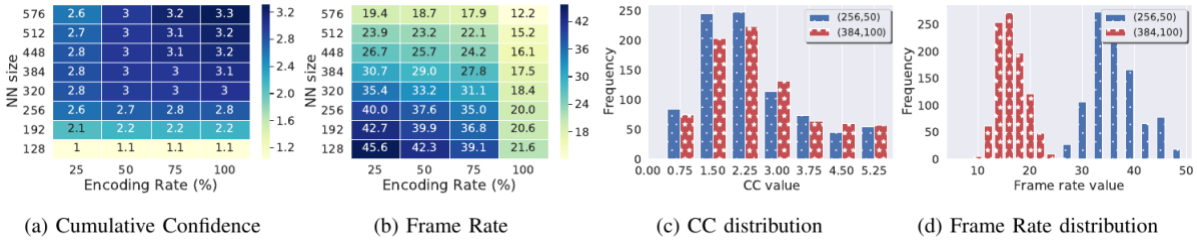
We assess the power behavior of the vBS by measuring the power consumption of its CPU and the whole baseband unit (BBU), the achieved performance in terms of throughput and goodput, details about the decoder at the vBS such as the subframe decoding time and the number of turbo decoder iterations per subframe, and some MAC and PHY indicators such as the Buffer Status Report (BSR), Block Error Rate (BSR), and the used modulation and coding scheme (MCS), and airtime. Moreover, we detect and identify unfeasible configurations in the dataset. This mainly occurs when an MCS value is forced but the channel quality is not good enough to decode its data.

3.3.5 Edge dataset (D5)

An Android application captures images through the mobile's camera, performs JPEG encoding, and transmits the compressed images to an edge server through a wireless 802.11ac Access Point. The collected data documents "latency" and "cumulative confidence" measurements that were obtained from this experiment. The records are obtained for different values of "image encoding rate" (i.e., compression) and "Neural Network input layer size" decisions. The delay is also provided with details on different phases: transmitting the frame wirelessly, decoding and rotating at the server, and performing object recognition with the state-of-the-art object recognition system YOLO on the server's GPU. YOLO accepts an image size that is a multiple of 32 and uses a number of potential object locations of different sizes to output a probability of each class in the training set to appear in each one. The extensive COCO

dataset is used, which covers a wide range of images and objects, and includes ground truth for each image. Moreover, the achievable frame rate as a result of the total latency is documented.

An analysis of the data (Figure 16) shows the effect of the chosen decoding and neural network size on the performance metrics we chose. Although these effects seem consistent in (a), (b), this is mainly due to taking the average across all images. In (c), (d), the variability in these metrics across the same configurations is demonstrated, motivating the necessity of tuning these parameters at run time.



(a) Cumulative Confidence (b) Frame Rate (c) CC distribution (d) Frame Rate distribution

Figure 16. *(a)-(b): Cumulative Confidence (CC) and frame rate for various neural network sizes and encoding rates; results are averaged across 32K images of the COCO dataset. (c)-(d): Distributions of CC and frame rate for (neural network size, encoding rate) set to (256, 50%), (384, 100%).*

To that end, this dataset includes 8 possible values for the NN size (128, 192, 256, 320, 384, 448, 512, 576) and 4 for the encoding rate (25, 50, 75, 100). For each NN size/encoding rate combination, the response of 1000 images -hence the dataset consists of 32000 records in total- from COCO dataset is tested and the latency response in each part of the process is measured, as well as the confidence values that are output from the DNN. The first 3 records of the dataset are presented in Table 6.

The first 2 columns of each record, are the "NN size" and "Encoding rate" respectively. They are followed by columns "Encoding", "Network", "Decoding", "Rotating", "YOLO", and "Server", which track the amount of time spent in milliseconds for each of these tasks, where "Server" is the total time spent on server processing including YOLO processing on the GPU. Column "Frame Rate" is simply the frame rate achieved for the image's end-to-end latency calculated by inverting the latter. The last 2 columns measure recognition performance and are the "Confidence" and "Cumulative Confidence". Confidence is an array of values in [0, 1], denoting the inference confidence for each identified object in the image. If no objects are recognized the array is empty. Cumulative Confidence is simply the sum of the Confidence array.

Table 6. *First 3 records of D5 dataset.*

NN size	Enc. Rate	Enc.	Network	Dec.	Rotating	YOLO	Server	Frame Rate	Conf.	Cum. Conf.
128	25	3	3.8	1.3	0.1	9.3	11.2	43.478	[0.863, 0,896]	1.76
128	25	4	6.3	2.3	0.1	10.8	14.7	38.462	[0.526]	0.526
128	25	4	5.7	1.5	0.1	9.2	11.3	43.478	[]	0

3.3.6 Wireless interactions in multiple BSS using Channel Bonding (D6)

This dataset [23] was created using Komondor [24], an open source, event-driven simulator based on the CompC++ COST library. Komondor is focused on fulfilling the need for assessing the novel features introduced in recent and future Wi-Fi amendments, which may be endowed with applications driven by techniques. The dataset was created in the context of the 2020 edition of the ITU AI/ML for 5G Challenge [25]. This dataset includes simulated data from IEEE 802.11 WLAN deployments applying Dynamic Channel Bonding (DCB). The dataset is divided into two parts, i.e., training and testing. In both cases, enterprise-like scenarios containing a different number of Access Points (Aps) and stations (STAs) applying DCB are generated, thus depicting multiple situations that could be used for training ML models. The topology of these enterprise-like scenarios is composed of a building floor of a given size (e.g., map size), which is divided in equal-sized offices. The Aps positioning is fixed, while the STAs are randomly placed around the AP coverage area. Such a topology setting is typically adopted in the simulation scenarios provided by IEEE 802.11 task groups [26]. The data set includes useful information about each deployment, such as the obtained throughput, the RSSI, the airtime in each channel, the interference among devices, or the SINR. In total, 600 deployments were simulated, containing 78,078 devices (namely, 6000 Aps and 72,078 STAs). Table 7 summarizes the main characteristics of the entire data set. Moreover, Table 8 details the simulation parameters used for generating the data sets.

Table 7. *Summary of the characteristic of the wireless interactions in multiple BSS dataset.*

Dataset	Name	Map Size (m^2)	Number of Deployments	Total Devices	Aps	STAs	AP Throughput [Mean, Std, Min, Max]	STA Throughput [Mean, Std, Min, Max]
Training/ Validation	1a	80x60	100	78,078: 6,000 Aps 72,078 STAs	12	[10-20]	[83.29, 52.24, 0, 400] Mbps	[6.93, 6.99, 0, 88] Mbps
	1b	70x50						
	1c	60x40						
	2a	60x40			8	[5-10]		
	2b	50x30						
	2c	40x20						
Testing	1	80x60	50	9,831: 1,400 Aps 8,431 STAs	4	Random	N/A	N/A
	2				6			
	3				8			
	4				10			

Table 8. *Simulation parameters used to generate the training and test datasets.*

	Parameter	Value	
		Training	Testing
Deployment	N° Aps	{8,12}	{4, 6, 8, 10}
	Aps Location	Fixed to the center of the cell	
	N° STAs	{5-10, 10-20}	5-10
	STAs Location	Uniform at random	
	Traffic profile	Downlink UDP	
	Traffic load	Full buffer mode	
	Channel Allocation	Uniform at random	
PHY	Central frequency	5 GHz	
	Path-loss model	See [27]	
	Bandwidth	{20, 40, 80, 160} MHz	
	N° Spatial streams	1	
	Allowed MCS indexes	1-12	
MAC	Contention Window	32 (fixed)	
	Data and ACK length	12000/32 bits	
	RTS and CTC length	160/112 bits	
	Max A-MPDU	1	
	DCB policy	Dynamic (see [28])	

Regarding the DCB configuration, each BSS can use up to N=8 basic non-overlapping channels of 20 MHz in the GHz band. Compliant with the 11ax amendment, a given transmitter can bond channels of width 20-160 MHz, thus leading to channelization C={{1},{2},⋯,{8},{1−2},{3−4},⋯,{7−8},{1−4},{5−8},{1−8}}, for basic channels indexed from 1 to 8. In each simulated deployment of the data set, both the primary and the total channel width are selected randomly. As for the applied DCB policy, the maximum possible channel width is dynamically used, provided that the involved channels were free during at least the point coordination function interframe space (PIFS) period. For instance, let us assume that a given transmitter has randomly allocated to channels {1−4}, with primary channel 1. Then, such a device would perform carrier sensing in the primary channel (1) and, provided that the channel was sensed to be free during the backoff, would assess whether the rest of the channels were also found to be free during the PIFS interval. If only channels {1−2} are idle at the moment of starting a transmission, then the transmitter proceeds to use both of them, leaving channels {3−4} for future transmissions.

The selected metric of performance is throughput, which is defined as the amount of successfully transmitted data in a period of time. Higher throughput is an indicator of higher bandwidth capacity (more bonded channels) but also interference-free communication. Nonetheless, Komondor can generate extensive and detailed performance statistics such as delay, spectrum utilization, or collisions. Moreover, the user can efficiently include as much as metrics as desired.

3.3.7 Intrusion Detection Evaluation Dataset (D7)

The Intrusion Detection Evaluation Dataset (CICIDS2017) is a validation dataset for anomaly-based intrusion detection approaches. It is provided open-source by the University of Brunswick (Canada) [29]. This dataset is used to evaluate the performance of machine learning algorithms for traffic classification implemented into the user plane. As one of the activity's targets is to fast respond to anomalies.

The CICIDS2017 dataset contains benign traffic and seven of the most up-to-date common attacks, which resembles the true real-world data, in PCAP format. Along with such traces, the dataset includes a list of all the flows labeled with time-stamp, source, and destination Ips, source and destination ports, protocols and attack, in CSV format. The traffic data contains all common available protocols, such as HTTP, HTTPS, FTP, SSH and email protocols. It also includes seven different types of attacks: brute force, Heartbleed, Botnet, DoS, Ddos, Web, Infiltration.

The dataset has been generated by means of a testbed where two networks: attack-network and victim-network. The former is a secure infrastructure with firewall, router, switches and a set of computers, running major operating systems, accompanied by an agent that implement different benign behaviors. The latter is a separated infrastructure with a router, a switch and a number of computers with public Ips that perform attacks toward the victim-network. Twelve and four different machines have been used for the victim-network and the attack-network respectively.

3.3.8 IPX Signaling Dataset for IoT (D8)

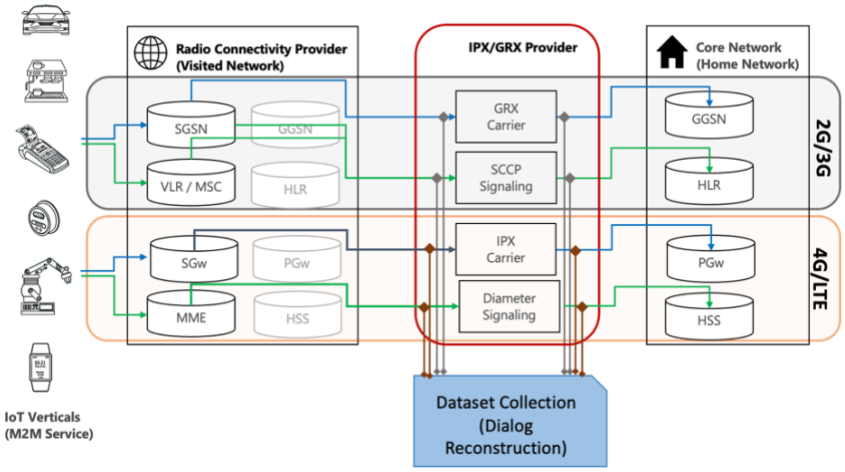


Figure 17. *High level architecture of the IPX-P's monitoring to build our dataset. We build our dataset using a commercial software solution that processes the raw signaling traffic (SCCP, Diameter or GTP), and that rebuilds the dialogues between the different core network elements. We build datasets for 2G/3G as well as 4G/LTE.*

We monitor the IP eXchange Provider (IPX-P) infrastructure that supports three core functions – Signaling Connection Control Part (SCCP) Signaling, Diameter Signaling, GTP signaling (for the corresponding radio technologies) – that enable the data roaming service for IoT devices. We show in Figure 17 a schematic view on the way we capture these corresponding datasets. We rely on a commercial software solution for capturing and analyzing in real time the raw signaling traffic, which we mirror from the signaling routers to a central collection point. In that central location, the commercial software re-builds the signaling dialogues between different core network elements in the visited and the home MNOs. Table 9 summarizes the datasets we use to characterize the operations of an IPX provider with a large international footprint.

Table 9. *IPX-P Datasets for IoT.*

Dataset	Infrastructure	Procedures Captured
SCCP Signaling	4 Signaling Transfer Points (STPs) (Miami, Puerto Rico, Frankfurt, Madrid)	MAP traffic, location management, authentication and security
Diameter Signaling	4 Diameter Routing Agents (DRAs) (Miami, Boca Raton, Frankfurt, Madrid)	Session Initiation Protocol (SIP) Registration, Voice over IP (VoIP) Call, Diameter Transaction, Domain Name Service (DNS) Query or RCS Session
Data Roaming	GTP-C control data and GTP-U data sessions.	Create/Delete Packet Data Protocol (PDP) Context/ Session; Flow-level metrics for data connections.
Ticketing System	Internal ticketing system for managing issues with the roaming platform.	Tickets information that track how an issue was handled by the operations team.

SCCP Signaling. This service provides the signaling capabilities for the second and third Generation (2G/3G) technologies. The IPX provider monitors the Mobile Application Part (MAP) protocol and specifically the traffic corresponding to the following procedures: i) location management (up-date

location, update GPRS location, cancel location, purge mobile device); ii) authentication and security (send authentication information); iii) fault recovery.

Diameter Signaling. This service provides signaling capabilities for 4G roaming. The IPX provider collects traffic corresponding to events including Session Initiation Protocol (SIP) registration, Voice over IP (VoIP) Call, Diameter Transaction, DNS Query or Rich Communication Services (RCS) Session.

Data Roaming. This service enables the data transport required for data roaming in 2G/3G and LTE. The IPX collect traffic related to the creation and management of GTP tunnels between roaming partners to transport data to and from end-users, as well as flow level metrics. Note that the service requires the use of the signaling platform, either LTE Diameter or SCCP signaling.

Ticketing System. This service keeps track of the incidents handled by the operation teams. Tickets may be generated by the operation teams, e.g., monitoring system alarms, or after an issue is reported by a customer. Every ticket contains a description of the incident and impacted devices or customers.

3.3.9 YouTube file requests (D9)

The open YouTube file request dataset [30] is a collection of traces from a campus network measurement on YouTube traffic. It contains trace data about user requests for specific YouTube content. The goal of this analysis is, first of all, to examine the local (I.e., popularity among videos requested in a trace file) and global (I.e., popularity information given from YouTube itself) popularity of YouTube video clips and furthermore, to accumulate real information from the traces, in order to be used in future simulations.

A part of this dataset is used in Activity A17 (Auto scaling Virtualized RAN caches), so as to generate traffic demand. The data utilized in our experiments is collected every 5 minutes for 7 days on a certain university campus. It is comprised of individual IDs of each requested video, requested time and destination/source IP addresses, video size, and transmission data rate. We also distinguish different regions with different IP addresses.

The aforementioned results are depicted in Figure 18. From there, it can be seen that the number of file requests have daily and weekday/weekend traffic patterns and more precisely, the traffic is high during the day and the evening until around midnight and then decreases significantly in the early morning hours. However, the traffic pattern is not similar even between two consecutive days, which makes the prediction of demand difficult.

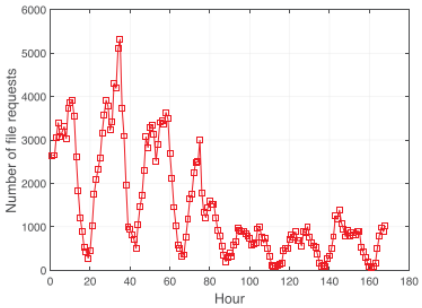


Figure 18. *Total number of YouTube file requests in a certain university campus over time.*

3.3.10 GEC case study (D10)

The GEC case study dataset is a collection of energy measurements taken from deploying a Generic Edge Computing (GEC) application that we have designed to represent a regular IoT/Edge/Cloud system [31]. We model the different elements of this deployment as a variability model with a large configuration space, in Figure 19. A variability model is a formal specification of the things that can vary in a system and how they depend on each other. Variability models are composed of features, which are elements or properties that can be in the final system or not. Features are organized in tree structures, so selecting a leaf for the final system depends on the selection of its ancestor features. In addition, it is possible to define explicit constraints, also known as cross-tree constraints. Features can be divided into 35 oolean features, numerical features, and variability sub-tree (aka clonable). Boolean features represent yes/no decisions or features that can appear in the final system or not. Numerical features are features that require a value to be resolved. Variability sub-trees mean creating instances or clones and providing per-instance resolution for features in its sub-tree. Cardinality features apply restrictions over the number of children of a feature that can be chosen. The appearance of a feature in a selection can be mandatory or optional. The GEC variability model comprises six main branches:

- Device: Edge computing hardware, as single-board and small appliances; in this evaluation, we consider four.
- Architecture: The microprocessor running architecture –commonly x86 and x86 64.

- Operating System (OS): The running OS in which the software will be executed, where, besides the top 100 UNIX systems in use in 2021 yearly published by Distrowatch 4, Microsoft Windows is considered. Please note that only the last available updated version and the default kernel were considered. Additionally, cross-tree constraints were defined for unsupported Oss.
- Programming Language (PL): The PL in which the operation is coded; in our evaluation, just the executed benchmarks are considered.
- Edge technology: Our available libraries and peripherals are considered, including wireless communication, data storage, temperature sensors, remote controllers, etc.
- Edge Context: Three key branches are located at this level. Parameters contain numerical features as natural numbers, usually input calculation parameters. Datatype represents the data types used in a specific operation. We used bytes for cases where several types are used simultaneously, like benchmarks. Operation contains the tasks performed in an IoT device; besides the common ones, such as starting or shutting down a device, the Phoronix Test Suite [32] operations are included in our evaluation. The suite ranges from battery power consumption monitoring for mobile devices to multi-threaded ray-tracing benchmarks and spans the CPU, graphics, system memory, disk storage, and motherboard components. While the suite comprises 403 tests, not all of them suits every OS or device of the variability model, but, on average, 100 are compatible for each system. As all the Oss, operations, and cross-tree constraints in Figure 19 are not graphically friendly, the complete ClaferVM is available [33].

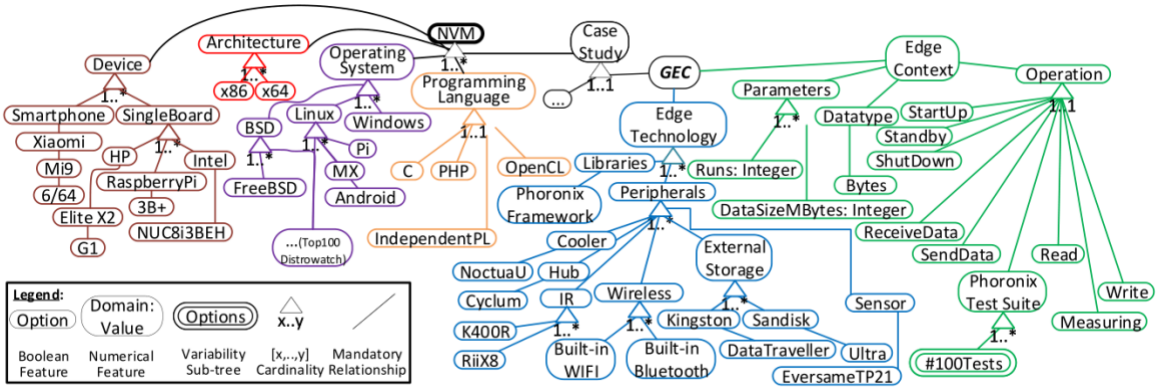


Figure 19. *VM of the Generic Edge Computing (GEC) large case study.*

Following the planning process for the energy-efficient software [69], we measured GEC energy consumption rate in Watts, triple checking with three professional tools: Watts UpPro Portable Power Meter, Multimeter Eversame C, and Eversame PowerMeter 2n1. The devices were highly cooled in a quiet and isolated room to avoid external factors affecting the readings.

Clafer chocosolver reasoner generated the ~5.3 $*10^8$ configurations of GEC in 36 hours for 552 Boolean and two numerical features with parent-children and cross-tree constraints. To approximate the GEC study to a real scenario, we performed 132500 different measurements, which account for 0.25% of the total search space – hence partially measured. TL confident scores were previously populated with 1,000 runs with random parameters (e.g., strategy, number of samples) and variability model constraints.

3.3.11 IoT devices dataset (D11)

UNSW-IoT [67] is a classification use case based on measurement data for 28 Internet of Things (IoT) devices, collected in a living lab emulating a smart environment. The objective is identifying the type of IoT device generating each traffic flow by looking at statistical features of the data packets. We employ 20 days of data that are made available to the research community, train the models over the first 15 days and use the last 5 days for testing.

3.3.12 Applications and protocols dataset (D12)

UNIBS-2009 [68, 69] is a traffic classification task based on real-world traces collected on the edge router of the University of Brescia campus network, capturing traffic from 20 workstations. The traces include web traffic (HTTP/HTTPS), mail (POP3, IMAP4, SMTP), peer-to-peer applications (BitTorrent, Edonkey) and other protocols (FTP, SSH). The goal is associating each traffic flow to one of 8 application or protocol categories. We use one day of traffic for training and a second for testing.

3.3.13 Malicious attacks dataset (D13)

NSL-KDD [70] is an anomaly detection case study that builds on 7 weeks of network traffic captured on a testbed recreating normal and attack traffic behaviors, by exploiting real hosts, live attacks and background traffic. The attacks fall in one of the following 4 categories: Denial of Service Attack (Dos), User to Root Attack (U2R), Remote to Local Attack (R2L), Probing Attack. The goal is again separating malicious and regular traffic. We consider the full data with the default training and test separation.

3.3.14 Malicious packets dataset (D14)

NSW-NB15 [71, 72] is an anomaly detection task employing a mix of real-world normal traffic and concurrent synthetic attack behaviors, produced in the Cyber Range Lab of UNSW Canberra. The measurements sum up to 100 GB of raw traffic with nine different types of attacks: Analysis, Backdoors, DoS, Exploits, Fuzzers, Generic, Reconnaissance, Shellcode, and Worms. Our goal, aligned with that of previous studies targeting this task, is once more to identify all malicious packets. In our experiments, we use 10 GB of data from the second day, and use 5 GB for model training and 5 GB for testing.

4 Results

As anticipated in Section 1, the results achieved by the project to date are organized into a set of specific activities, which jointly contribute to progressing in each planned evaluation. ***Overall, the project carried out 23 activities to date. In the following subsections, we present the results for each evaluation E1-E7, and detail the outcome of each associated activity.***

At the start of each subsection we also summarize in a table the activities performed, also reporting the tools from Section 3 employed by each activity, the target and collected KPIs, the target Technology Readiness Level (TRL) and either the activity is planned or not for the PoC demo. Finally, the last column of summary tables provides an indicative figure of the current progress towards the completion of the activity. We also report in the same tables the main innovations entailed by each activity.

We remark that neither all evaluations have the same volume of activities, nor all activities have the same level of progress. This was planned since the beginning, and it is due to the fact that (i) evaluations target network functionalities with diverse levels of maturity and complexity, hence requiring a more or less intense effort by the project, and (ii) activities employs very diverse NI models and tools, whose development over time cannot be uniform or perfectly aligned. In all cases, all evaluations have achieved some preliminary results to date: in some cases, those consist in the development of the platform needed to run experiments, in some other in a data-driven assessment of the problem, and in others yet in a first assessment of actual NI-assisted solutions. We expect all evaluations and activities to better align during the second iteration of the project, in the sense that they will all produce early or consolidated results about the performance of the NI-assisted solutions. This will be captured in the following deliverable of WP5.

4.1 NI for sustainable virtualized RANs

Evaluation E1 focuses on real-time control and non-real-time orchestration of vRAN services & resources. The DAEMON consortium performed assessments of challenges and solutions related to E1 via activities A1-A5. Table 10 summarizes the tools, KPIs, TRL, PoC plans, approximate progress and main innovations of such activities.

Table 10. *List of activities for E1.*

ID	Name	Evaluation	Tool	Planned KPIs	Collected KPIs	Target TRL	Planned for PoC demo	Progress
A1	Reliable RAN virtualization	**E1**	T1	K1, K2, K4, K5	K2, K4, K5	4	Yes	30%
	Main innovation: 1) Reliable DU design suitable for Virtualization; 2) Centralized real-time control of radio and computing resources							
A2	AI-driven O-Cloud	**E1**	T1	K1, K2, K4, K5	K2, K4	4	Yes	30%
	Main innovation: Centralized real-time control of radio and computing resources							
A3	Application aware radio scheduling	**E1**	S2	K9	K9	3	TBD	30%
	Main innovation: Traffic classifier to steer scheduler settings							
A4	AI-aided energy-driven RAN orchestration	**E1**	T5	K1, K4	K1	4	Yes	50%
	Main innovation: Energy-driven O-RAN orchestration (non-real-time RIC)							
A5	AI-aided RAN/edge orchestration	**E1**	T5	K1, K4	K1	4	Yes	50%
	Main innovation: Joint Energy-driven O-RAN orchestration (non-real-time RIC) and AI service							

Overall, the preliminary results of these activities already led to a number of observations on the use of NI for virtualized RANs in next-generation mobile systems, as follows.

- ***We identify gaps in the current implementations of vRANs, in terms of usage of shared resources across pools of Distributed Units (DU)***. This occurs both due to (i) the lack of guarantees of a timely completion of DU jobs when the processing times are non-deterministic, and (ii) substantial economic and energy costs of hardware accelerators (Has) used in such shared resources. These issues are explored in ***A1*** and ***A2*** below. Our results call for (i) a re-design of the DU pipeline, and (ii) NI-driven approaches for operating vRANs so as to limit the need for Has to a minimum, which will be designed and implemented during the second iteration within the project, and whose performance assessment will be presented in the next deliverable of WP5.
- ***We implement and compare different strategies to address the currently open problem of traffic classification in vRAN***, using local information about the channel bearer and radio link buffer. In ***A3***, we show that a simple heuristic and a complex deep learning models perform similarly in this task, and can thus inform solutions for traffic-aware radio scheduling. As there is still space for improvement in the absolute performance of these classifiers, the second iteration of the activity will focus on further refining the solutions, and results will appear in the next deliverable of WP5.
- ***We reveal that power consumption of virtualized Base Stations (vBS) in vRANs is in practice much more complicate than what assumed in the literature, and that it is linked to end-user QoS in introcated ways***. Our results are obtained via experiments in ***A4*** and ***A5***, using a dedicated real-world platform, and show tangled relationships of traffic, SNR, MCS and airtime, with non-linear and non-monotonic relations between system configurations and power usage or attained throughput. Similarly complex relations exist with application-level performance metrics such has the latency and throughout. While all these results clearly hinder the derivation of general consumption models and the use of simple decision models to drive resources, we plan to devise NI algorithms that can cope with such complexity in the second iteration of the project.

4.1.1 Reliable distributed unit for virtualization (A1)

As discussed in Section 4.1 of [34], achieving very high reliability for virtualized RAN on a shared cloud platform is one of the challenges of next generation mobile networks. In this Section, we provide some evidence of the fact that the current architecture of the mobile network u-plane pipeline is not optimized for its usage in a cloud computing environment.

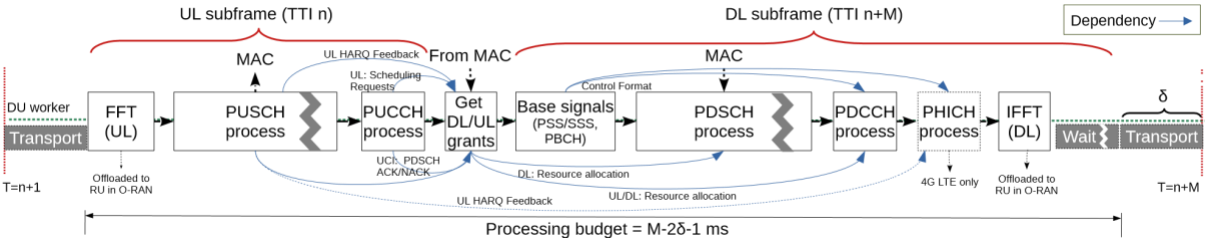


Figure 20. *LTE and New Radio (NR) DU pipeline: DU job n.*

4.1.1.1 Reliability of the baseline pipeline

The design of the baseline pipeline, described in section 4.1.1.2 of [34] and shown in Figure 20, is not suited for non-deterministic computing platforms such as shared clouds. Namely, existing solutions implementing the above baseline pipeline cannot guarantee the timely execution of individual jobs without the assistance of dedicated hardware acceleration or aggressive over-dimensioning [35], which compromise flexibility and cost-efficiency [36].

4.1.1.2 Timing Constraints

3GPP defines several timing constraints [37]. Relevant to this discussion are C3 (the latency between ACK/NACK reception in Uplink (UL) UCI and the corresponding Downling (DL) re-transmission in PDSCH), and C4 (latency between PUSCH reception and delivery of HARQ feedback). In LTE, C4 = 3 ms, which implies $M=4$. Though these timings are more flexible in NR, they are set at longer timescales by the CU, however keeping the same choice [38]. As a consequence, there is a hard deadline to process each DU job within $M-1$ ms, as shown at the bottom of Figure 20. Violating this deadline prevents timely delivery of DL SFs and, as a result, loss of synchronization and connectivity between the DU and its users, as shown by the baseline performance in the experiment of Figure 21.

4.1.1.3 Inter-task dependencies

Regardless of individual processing optimizations [39], different DU tasks within a job have strong dependencies as shown by the blue arrows in Figure 20. DL grants must be computed before PDSCH because they carry information required to encode and modulate DL TBs. As a result, known

implementations (e.g., Samsung's vDU [40], srsRAN and OpenAirInterface) perform each DU job in a single thread pipeline (Figure 20), or in a multi-thread pipeline where each thread has to wait and be executed in a precise order [41], which boils down to Figure 20 again. Although solutions like Agora and FlexRAN help to accelerate the processing of individual DU tasks, the aforementioned dependencies prevent running different DU tasks in a job in parallel to expedite the pipeline of Figure 20.

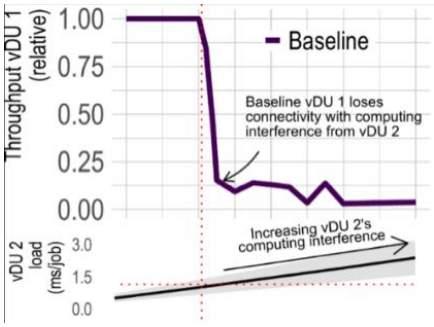


Figure 21. *Throughput measured for two vDUs competing for resources.*

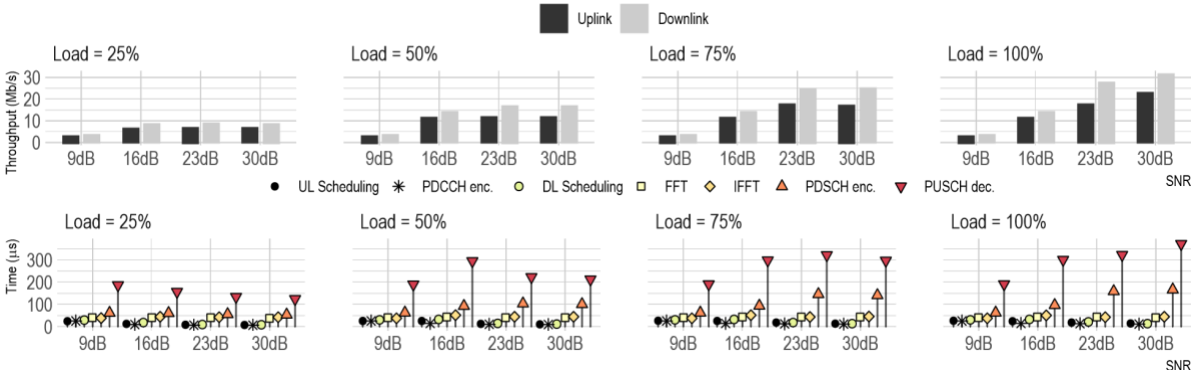


Figure 22. *Throughput performance for both uplink and downlink (top). CPU time required by different PHY layer functions (bottom). Different uplink/downlink load (relative to the maximum) and channel conditions (SNR).*

4.1.1.4 Non-deterministic tasks

As hinted in our toy experiment shown in Figure 21, the computing time required by DU tasks highly depends on the instantaneous availability of computing resources. We note moreover that the most compute-intensive tasks also depend on the context, that is, on the data load (rate of TBs to decode/encode) and on the mobility patterns of the users (signal quality) [42], which can induce very quick fluctuations in the demand for computing resources. To illustrate this, we deploy the baseline vDU, implemented in srsRAN, processing downlink and uplink traffic over one Intel i7 core in a 10-MHz band. Figure 22 depicts the achieved throughput in both the uplink and downlink (top subplots), and the median time incurred by the CPU to perform DU tasks (bottom subplots). We take these measurements for different load intensities (relative to the capacity in UL and DL, respectively) and average signal-to-noise ratios (SNR) indicating the channel quality for both UL and DL, and adapt the MCS to minimize the workload issued by the decoder, differently to our results. The results yield two observations. First, processing PDSCH and (especially) PUSCH are the two tasks that consume CPU time the most, which is not surprising as it has been observed before [43, 44]. Second, while the CPU time of the rest of tasks (and others not shown in the figure to reduce clutter) remains practically constant, the time required to process PDSCH and PUSCH highly depends on the context; that is, on the SNR—and so on the mobility patterns of the users, and on the load—and hence on the users behavior. Note that even if shared pools of hardware accelerators are used *à la cloud* to reduce the processing time of some of these tasks, queueing in the abstraction layer brokering access to the accelerators across multiple vDUs incur in similar issues [45].

4.1.1.5 Conclusion and outlook

Because of the above, baseline solutions cannot guarantee the timely completion of DU jobs when facing computing fluctuations, which cause unreliability in scenarios such as that of Figure 20. We hence claim that a re-design of the DU pipeline is required for non-deterministic computing platforms such as shared clouds, which we will fully describe in the next version of the WP5 deliverable.

4.1.2 AI-driven O-Cloud (A2)

vRAN solutions in the market today resort to offloading compute-intensive tasks, usually FEC operations, into *dedicated* hardware accelerators (HAs). HAs are GPUs, FPGAs, or even CPUs that are programmed to perform one single task. Because they are programmable and can carry out FEC LDPC work very fast, GPUs (NVIDIA Aerial) and FPGAs (Intel FlexRAN[3]) are the most popular HAs for 5G vRANs. For instance, Figure 23 shows that a 10x latency reduction can be achieved when offloading LDPC workload into a GPU. In line with the industry, we focus on FEC offloading in this activity.

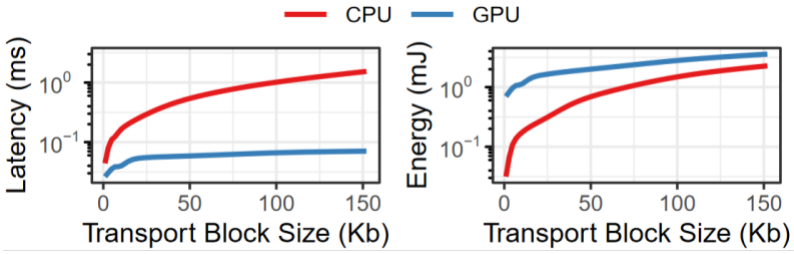


Figure 23. *Mean latency and energy consumption to decode an LDPC-encoded transport block.*

4.1.2.1 Limits of hardware accelerators for vRAN

However, as acknowledged by top executives in the business, this approach is doomed to fail. The root cause is the strong dependency on HAs, which are more expensive and energy-inefficient than ASICs (see Figure 24) yet are dedicated to individual DUs. Notably, the energy toll of a GPU-based LDPC decoder has an average consumption of 8.25 nJ per bit per decoding iteration. In marked contrast, ASIC-based decoders consume as low as ~3 pJ/bit/iteration.

This is a waste because (i) HA resources are highly underutilized most of the time, and (ii) less-performing yet low-cost processors are not sufficiently exploited to bear this workload. Note in Figure 23 (right) that CPUs can process FEC operations with a 6x lower energy toll.

	CPU	GPU	FPGA	ASIC
Time-to-market (months)	<2	<2	2	30
Unit cost ($)	180†	10000†	4000†	30
NRE* cost ($)	0	0	0	350K-1500K
EE$^{\Psi}$ (pJ/bit/iter)	1400†	8250†	450	3

*Non-Recurrent-Engineering

$^{\Psi}$Energy Efficiency when decoding LDPC codes.

†Intel Xeon 6240R CPU, NVIDIA V100 GPU and Intel PAC N3000 FPGA as of Dec. 21.

Figure 24. *Comparison of HAs. Approximated figures.*

4.1.2.2 Conclusions and outlook

Our current measurements reveal that HAs are not necessarily the silver bullet for vRAN functionality operation, due to the substantial capital expenditure and energy footprint they entail. This motivates the need for NI-assisted solutions that can take these costs in the equation, and take vRAN operation decisions (e.g., allocating bandwith to users associated to specific base stations) that optimize the utilization of (limited) HA resources, thus limiting the need for deployed HA capacity. We will explore these challenges and propose solutions in the next WP5 deliverable.

4.1.3 Application aware radio scheduling (A3)

With the simulator S2, described in Section 3.2.2, we generated data to assess the performance of the classification algorithm described in Section 4.3 of Deliverable D3.1 [34]. Typically, we let a number of mobile users, say 8, consume[4] data under the form of a large file transfer, streaming video, web browsing or Internet radio and observe the air interface. Specifically, we observe the (downlink and uplink) throughput on the bearer and the (downlink and uplink) evolution of the RLC (radio link control) buffer. A typical example of a case where mobile 1 and 2 consume Youtube video, mobile 3 and 4 are web-browsing, mobile 5 and 6 are consuming Internet radio and mobile 7 and 8 are downloading a large file, is shown in Figure 25. Note that the granularity of the RLC buffer occupancy, which has a granularity of 10ms, is 100 times finer than that of the throughput, which has a granularity of 1 sec.

[3] Intel FlexRAN provides both FPGA drivers and CPU libraries (exploiting Intel AVX-512 instruction sets) for FEC acceleration.

[4] For now, we consider only applications that mainly download information, but later we may consider applications that produce information as well.

Based on such traces acquired in various conditions with various types of mixtures, we construct labeled data as follows. Each trace (for which we know the nature of the traffic that is being generated) is split in chunks of a finite size (say 20 sec) and that chunk inherits the label of the trace is belongs to. In that way we build up a data base of labeled chunks with which we can train, validate and test any classification algorithm that we care to investigate. We split the data set (randomly) in a training set that comprises 75% of the labelled data and a test set that consists of the remaining 25%.

To illustrate the performance of the classification algorithms we show the confusion matrix calculated on the test set, i.e., entry $cm[k,l]$ of the confusion matrix is the fraction of cases of class k that we classified as class l. Ideally, we want the diagonal entries of the confusion matrix to be close to 1. Notice that each row of the confusion matrix sums to 1.

Here we discuss in more details two of the three classification algorithms introduced in Section 4.3 of Deliverable D3.1 [34] that take 20sec (i.e., 2000 samples) of the RLC buffer occupancy as input vector and produce an application category as output. There are four categories: file transfer, video, web browsing and internet radio, and from the traces we extracted 20880 vectors of the file transfer category, 9180 vectors of the video category, 6869 vectors of the web browsing category and 6964 vectors of the Internet radio category.

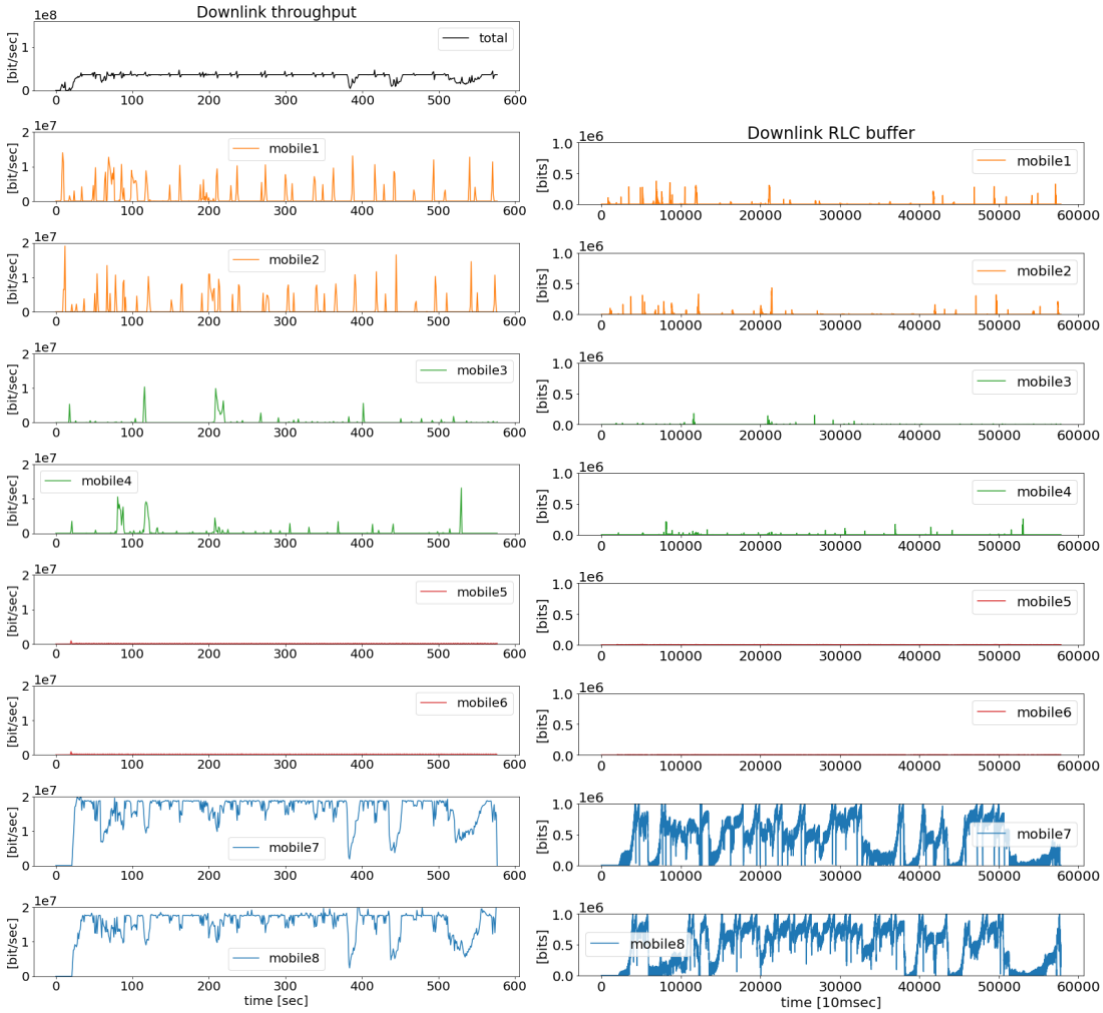


Figure 25. *Downstream throughput (left) and RLC buffer (right) evoluation in a typical experiment.*

4.1.3.1 K nearest neighbors algorithm

When an input vector is presented to this algorithm, it determines the K closest neighbors in the training set, then it determines the labels that are associated to these neighbors and takes a weighted majority vote to determine the class associated to this input vector. In the results below, we choose Euclidean distance as distance metric, K = 4 and the weights are the inverse of the distance.

Table 11 shows the confusion matrix of this classifier. File transfer and video can be easily identified. However, web is often misclassified as video and vice versa, while radio is misclassified very often, most of the time as video. Since radio does not consume too much bit rate this is not a major problem.

Table 11. *Confusion matrix of K nearest neighbors.*

		Classified as:			
		File transfer	Video	Web	Radio
Ground truth	File transfer	0.964	0.027	0.003	0.007
	Video	0.013	0.837	0.149	0.001
	Web	0.013	0.22	0.766	0.001
	Radio	0.002	0.112	0.591	0.295

4.1.3.2 *Feed forward neural network*

We chose a neural network with an input layer of 2000 neurons and an output layer of as many neurons as there are traffic classes. We considered 3 hidden layers with 40 neurons each. The activation function for all neurons, except the output layer, is ReLu, where $ReL(x) = x$ if $x > 0$ and 0 otherwise. The output layer has softmax, where $softmax(x_k) = \exp(x_k)/\sum_i \exp(x_i)$, as its activation function and can be interpreted as the probability that the presented input vector belongs to class k. As loss function we used categorical cross entropy and we used ADAM as training algorithm. Of the training set, we set 20% of the data aside for validation. We trained, with a batch size of 32 samples, as long as the loss on the validation set substantially decreases, i.e., as long as it does not increase 10 epochs in a row, as illustrated in Figure 26.

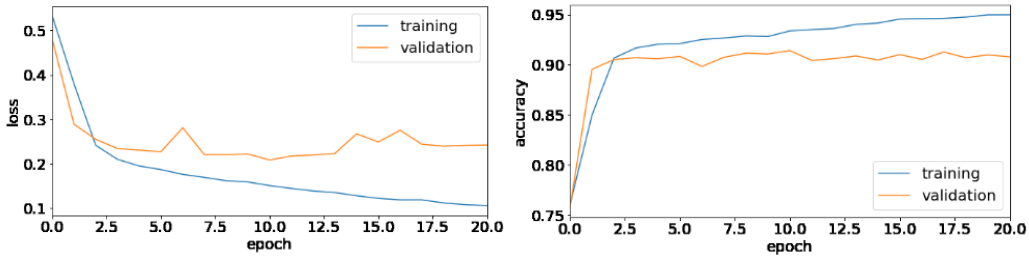


Figure 26. *Evolution of the loss function and classification accuracy on the training and validation set.*

Table 12. *Confusion matrix for the neural network classifier.*

		Classified as:			
		File transfer	Video	Web	Radio
Ground truth	File transfer	0.992	0.006	0.002	0.000
	Video	0.017	0.890	0.091	0.001
	Web	0.001	0.364	0.631	0.004
	Radio	0.008	0.003	0.020	0.969

Table 12 shows the confusion matrix for this classifier. Again, file transfer is easily identified. Video is confused with web more often with this classifier than with the K nearest neighbors classifier, while on the contrary web traffic is classified better. Radio is nearly always classified correctly.

4.1.3.3 *Conclusions*

The two classifiers perform similarly. Classification based on a chunk of 20 sec in a heterogeneous environment is not trivial. Since the length of the chunk is also the (minimal) response time of the classifier, it cannot be taken it much longer. In future work we will investigate if a post-processing step, where we take into account how chunks that immediately precede the chunk to be classified are classified, can help. Such an analysis will be the focus of the second iteration of the activity, and its results will be reported in the next deliverable of WP5.

4.1.4 AI-aided energy-driven RAN orchestration (A4)

This NI-solution is outlined in Section 3.2.1 of Deliverable 4.1 [46]. We used testbed T5 to perform our evaluation. In this section we summarize the most important findings.

4.1.4.1 *BBU/CPU Power Cost & Impact of Platform*

The first important finding is that the power consumption associated with BBU processing is *comparable* to the RFR chain's transmission power. This result is consistent with previous studies. Figure 27 dissects the power consumption of a vBS deployed over a Small Factor (SF) PC into the share responsible by (i) the BBU´s CPUs[5], (ii) the BBU´s cloud platform *except the CPUs*, and (iii) the actual RU deployed over a Software Defined Radio (SDR) transceiver from National Instruments. We measure the power

[5] We use the Intel's Running Average Power Limit (RAPL) functionality integrated into the Linux kernel to measure the CPU consumed power.

consumption for four scenarios: (i) the vBS is not deployed (baseline), (ii) the vBS is deployed with an idle user attached (vBS idle), (iii) the vBS is transmitting 20Mbps of DL traffic, and (iv) the user is transmitting 20Mbps of UL traffic to vBS.

Excluding the baseline scenario, the CPU power cost alone is, on average, 29% larger than that of the RU, while the overall BBU power exceeds it by 175%, on average (208% over full UL load). Interestingly, these numbers depend on the platform, which hosts the BBU. Namely, Figure 28 shows the BBU consumption over the baseline for different platforms.[6] We compare the power consumed by the BBU in idle state and operating at full UL/DL buffer and subtract the baseline power. Indeed, the power cost changes significantly, and is affected also by the vBS bandwidth.

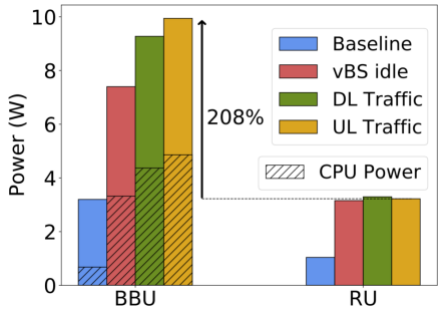


Figure 27. *Comparison of power consumption at: the BBU (Intel NUC i7-8559U@2.70GHz), the BBU's CPU, and the RU (an USRP SDR), with 20Mbps DL and UL traffic.*

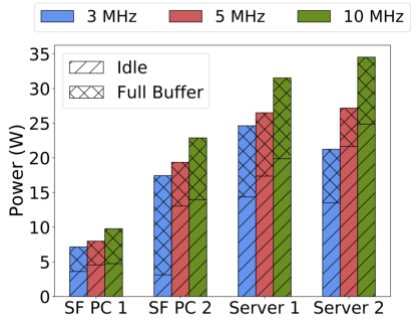


Figure 28. *Consumed power over the baseline for different radio bandwidths and hardware platforms. SF PC 1: Intel NUC i7-8559U@2.70GHz; SF PC 2: Intel NUC i7-8650U@1.90GHz; Server 1: Dell XPS 8900 i7-6700@3.40GHz; Server 2: Dell Aurora R5 i7-9700@3.00GHz.*

4.1.4.2 *Impact of SNR & MCS*

The second finding is that the SNR of the wireless channel and the MCS in UL affect the BBU computing load, and hence, its power consumption in a non-linear fashion. This is because the decoder needs more iterations when the received signal becomes noisier. Thus, the decoding time per subframe increases, e.g., by 52% between 20 and 15 dBs for MCS 23, see Figure 29; and this induces a commensurate increase in power consumption, see Figure 30. Besides, Figure 30 shows that, even for a given decoding time, higher MCS values induce more power consumption, which is attributed to their more intricate demodulation. Importantly, excessive decoding delays can induce throughput loss since they lead to violations of vBS deadlines [47]. Hence, maximizing throughput does not only have an unpredictable effect on power, but it is indeed highly non-trivial.

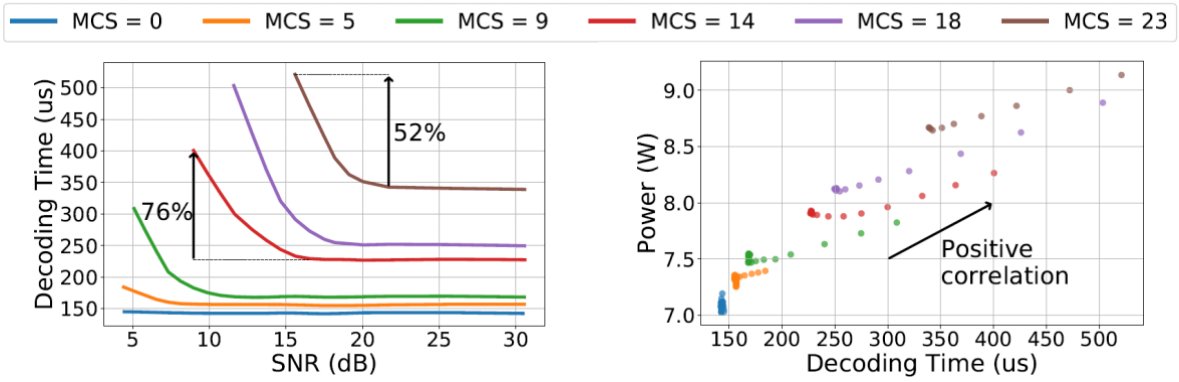


Figure 29. *vBS over SF PC 1 at full UL buffer. UL decoding time as a function of SNR and different MCS values.*

Figure 30. *vBS over SF PC 1 at full UL buffer. Power consumption as a function of the decoder performance (high correlation).*

[6] The small factor PCs consume less power than the servers, which however can host more vBSs hence are expected to consume less power per user.

4.1.4.3 Configuration Options & Impact of Scheduler

The above vBS control challenges are exacerbated by the plenitude of configuration options. Figure 31, for instance, presents combinations of MCS and airtime values (percentage of used subframes) achieving the same UL throughput. Configurations with higher MCSs (and therefore lower airtime) reduce power by 38%. However, this relation is *non-monotonic*, as we have also measured higher power when the MCS increases and SNR is relatively low; this is due to the fast increase of computing load (see Figure 30). On the other hand, configurations 6 to 8 have the same power consumption, but still differ since configuration 8 involves lower airtime and thus can serve more users, while configuration 6 is more resilient to noise. These decisions are made by the vBS radio scheduler which based on the SNR selects the MCS and airtime. Figure 32 shows the power consumption as a function of MCS and airtime for UL transmissions. We observe that both parameters have a smooth impact on power, but in practice this characterization is not available and needs to be *learned*.

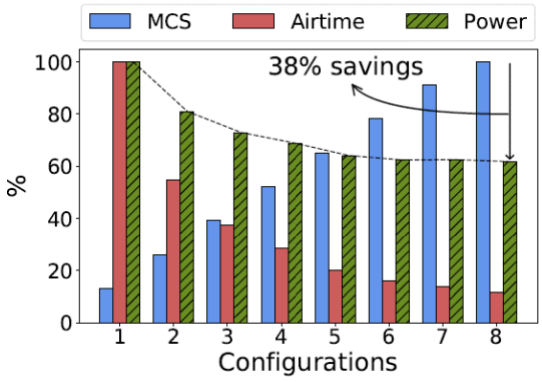


Figure 31. *8x combinations of normalized MCS and airtime providing 2.6Mbps in UL, and its associated power (idle mode power is subtracted).*

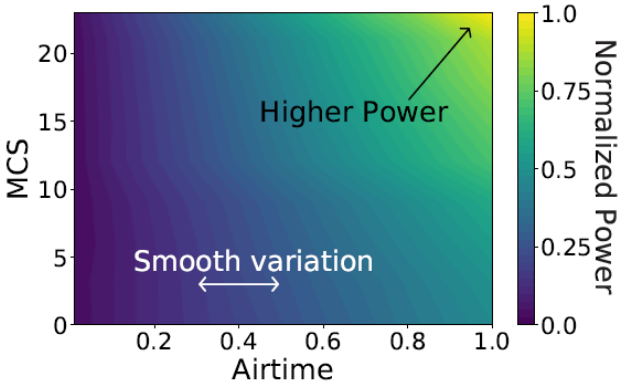


Figure 32. *Normalized power consumption at the BBU over baseline for full buffer UL transmissions and high SNR, as a function of MCS and airtime.*

4.1.4.4 Conclusions

Characterizing the vBS power consumption is intricate as it depends on traffic, SNR, MCS and airtime. There are many DL and UL configurations and some of them present *non-linear and non-monotonic* relations with power and throughput. Moreover, the power consumption depends on the BBU platform and radio scheduler. This hinders the derivation of general consumption models.

4.1.5 AI-aided RAN/edge orchestration (A5)

This experimental activity targets the performance assessment of the solution for NI-assisted RAN/edge orchestration reported in D4.1, Section 3.3.1 [46]. Here, we summarize the main findings of the evaluation.

4.1.5.1 GPU-enabled Edge server for mobile video analytics

We have built a fully-fledged prototype system with a software-defined BS (using srsRAN suite) and a GPU-enabled edge server that offers a Mobile Video Analytics (MVA) service to mobile users. We measure the joint impact that resource control policies at the user device (frame size), the BS (radio configuration) and the server (GPU speed) have on the service accuracy and end-to-end latency (QoS), and on power consumption (cost). Our experiments show that, unlike other services, performance is highly volatile and depends on the underlying hardware, the AI service configuration, and even the actual user data. Furthermore, these services include a wide range of configuration options, e.g., selecting different architectures of neural networks, different processing equipment, or even adjusting the data sources. All these parameters affect in an unknown way the latency and accuracy, which in turn renders traditional resource orchestration techniques ineffective for this problem.

We have performed an exhaustive set of experiments using a testbed. In a nutshell, the testbed is comprised of a 3GPP R10-compliant LTE BS, a UE generating service requests via the BS to a well-known object recognition service, and an off-the-shelf server with an NVIDIA GPU running the service. Each request consists of an image with a variable number of objects from the COCO dataset[7]. The images are sent to the service via the uplink channel of the LTE interface, and the service returns to the user a bounding box and a classification label for each identified object in the image. This information is sent via the downlink channel of the LTE interface. Each measurement shown as a dot in the figures of this section is an average of 150 images. The dataset collecting all the measurements shown in this section is

[7] https://cocodataset.org/

available online[8] to enable reproducibility and to facilitate further research in this area. In the following, we analyze the trade-offs between different *configuration policies* and *performance indicators* that are relevant to the system *stakeholders*: (i) quality of service experienced by the end-users, (ii) the cost associated with the service provider, and (iii) the cost associated with the MNO.

4.1.5.2 Latency and precision

We start off by analyzing two metrics of interest for the user's quality of service: the service's performance to recognize objects and the service delay, formally introduced in Performance Indicator 1 (Service Delay) and 2 (Mean Average Precision), respectively. According to our measurements, the most relevant feature that affects the mAP is the *image resolution*, defined in Policy 1.

We illustrate this in Figure 33, which shows the trade-off between service delay and mAP for the COCO images dataset encoded with different resolutions. The remaining configuration policies (described later) are fixed so the service delay is minimum. The results are rather intuitive: (i) Higher-resolution images carry more pixels encoded in a larger amount of data. Therefore, higher-resolution images incur higher delay due to longer transmission time over the radio interface. (ii) Lower-resolution images cause the service to provide lower mAP performance because they carry less useful information for the object detection engine. Specifically, in our experiments, a 72% improvement in service delay is associated with a reduction of precision that goes between 10% to 50%.

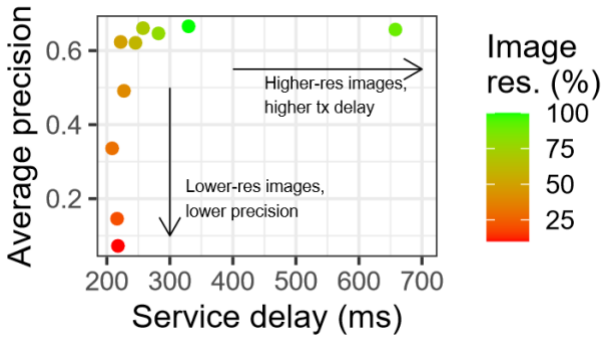


Figure 33. *Mean average precision (mAP) vs.service delay for images with different resolutions.*

4.1.5.3 Including power consumption in the picture

There also exists a trade-off, which naturally appears in many resource control problems, between the users' QoS and the associated cost to the provider of such service. To explore this trade-off, we introduce a policy that governs the allocation of radio resources, defined as Policy 2, and an additional metric that assesses part of the aforementioned cost: the server's power consumption, defined as Performance Indicator 3.

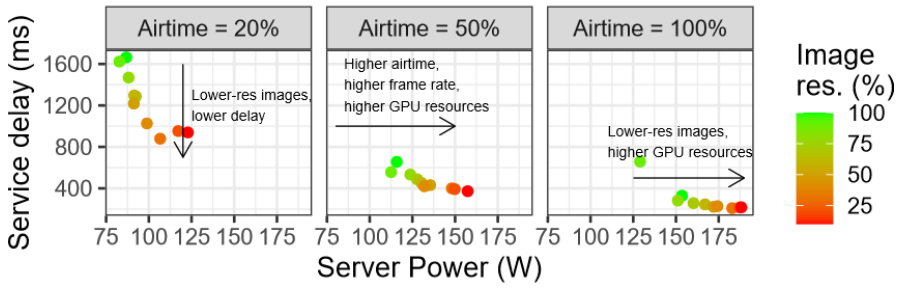


Figure 34. *Service delay vs. server's power consumption for images with different resolutions and radio policies.*

Figure 34 depicts the service delay versus the server power consumption, for different airtime radio policies and image resolutions. Similarly, as before, higher resolution images increase service delay due to the longer transmission time of the requests. We now observe that this occurs irrespective of the radio policy configuration. However, the selected radio policy has an important impact on service delay as well, which is intuitive as lower airtime implies lower usage of radio resources, which further increase the transmission time of the requests at the radio interface. Specifically, our experiments show that an 80% increase of the airtime produces improves the delay between 65% and 80%. Concerning the server's power consumption, lower resolution images and lower radio resource allocations increase this cost for

[8] https://github.com/jaayala/

the service provider. Specifically, there is a 56% increase in power consumption for an 80% increase in spectrum time resource; a similar increase attained when there is a 75% increase in image resolution. This is explained because larger amounts of radio resources allow the user to send a higher rate of requests in a similar way than lower resolution images do, which ultimately increase the amount of work assigned to the service's resources (the GPU in this case).

4.1.5.4 Conclusions

The relationships between key performance indicators in the context of MVA in a GPU-enabled Edge server are entangled and offers a number of trade-offs. Note that additional trade-offs to those above are also detailed in D4.1 [46], Section 3.3.1. Clearly, simple decision-making models cannot capture such complex relations, which paves the road to the development of dedicated date-driven NI algorithms that can effectively learn such intricated correlations. This will be the objective of the activity during the second iteration of the project.

4.2 NI for VNF placement and control

Evaluation E2 focuses on NI solutions that support network slice management & orchestration operations. The DAEMON consortium performed assessments of challenges and solutions related to E2 via activities A6-A8. Table 13 summarizes the tools, KPIs, TRL, PoC plans, approximate progress and main innovations of such activities.

Table 13. *List of activities for E2.*

ID	Name	Evaluation	Tool	Planned KPIs	Collected KPIs	Target TRL	Planned for PoC demo	Progress
A6	Energy-aware deployment of VNFs for genenric Edge computing	**E2**	D10	K1	K1	3	TBD	10%
	Main innovation: Optimization of edge computing system deployments exploiting variability							
A7	Combining VNFs at the edge	**E2**	None[9]	K1, K2	K1, K2	3	TBD	10%
	Main innovation: Adaptation of VNFs resource requirements to the available Edge resources							
A8	AI Enhanced MANO	**E2**	T8	K3, K8	K3	5	Yes	30%
	Main innovation: Automated resource reallocation for supporting critical services							

Overall, the preliminary results of these activities already led to a number of observations on the management and orchestration of network slices in next-generation mobile systems, as follows.

- ***We contribute to the challenging endeavor of making VNF placement and control more energy friendly in complex mobile Edge settings***, where the configuration space is extremely large and network services developers can measure the energy consumption of only a subset of these configurations. Specifcally, activities ***A6*** and **A7** tackle this problem by (i) developing an interactive and statistical approach to provide energy consumption insights based on a small subset of directly accessible measurements, and (ii) discovering and adapting Edge resources to the VNFs requirements in an energy-aware fashion, respectively.
- ***We developed a complete framework for NI-assisted MANO***. By engineering together a number of components, in activity ***A8*** we deployed an experimental platform that will support, in the rest of the project lifetime, real-world tests of NI instances that can automate MANO operations in next-generation mobile networks. The actual implementation and integration of such instances will be carried out during the second iteration of the project, with results presented in the next deliverable of WP5.

[9] Activities A7 makes use of randomly generated data to validate the solution.

4.2.1 Energy-aware deployment of VNFs for genenric Edge computing (A6)

In this Section we report the results of the evaluation of the SAVRUS (Smart Analyser of Variability Requirements of Unknown Spaces) approach presented in Section 5.3.2 of deliverable D3.1 [34]. The main objective of SAVRUS is to guide in the optimization of Edge computing system deployments by identifying the specific features with a high probability to improve the energy consumption if they are replaced by their alternatives.

We have performed a proof of concept of SAVRUS using an Edge computing case study. In addition, we have evaluated and compared the two implementations of our approach, using the product sampling techniques Diversified Distance-based Sampling (DDbS) and Statistical Recursive Search (SRS). The results are shown in Table 14 and discussed below. Our evaluation is based on variability models and the dataset D10 presented in Section 3.3.10.

Table 14. *Details of the quality measured numerical variability models (NVM) used to validate SAVRUS.*

NVM	Description	#Boolean	#Numericals	Space	QA	#Measurements
Dune	Muti-grid solver	11	3	2,304	Equation solving time	2,304
HSMGP	Stencil-grid solver	14	3	3,456		3,456
HiPAcc	Image processing framework	33	2	13,485		13,485
Trimesh	Triangle mesh library	13	4	239,360		239,360
GEC	Generic edge computing	**552**	2	**~ 5.3 * 10^8**	Energy Consumption	**132,500**

4.2.1.1 Generic Edge computing case study

Firstly, we tested SAVRUS with a Generic Edge Computing (GEC) case study – a variability model with a large configuration space that we have designed to represent a regular IoT/Edge/Cloud system for VNFs (described in Section 3.3.10). Its main details are shown at the bottom of Table 14. The clafer chocosolver reasoner[10] was used to generate the $\sim 5.3 * 10^8$ configurations of GEC in 36 hours for 552 Boolean and two numerical features with parent-children and cross-tree constraints. GEC search space details are also specified in the last row of Table 14, being the largest space considered to evaluate this work. To approximate GEC study to a real scenario, we performed 132,500 different measurements, which account for 0.25% of the total search space – hence partially measured. We found interesting feature interactions and optimization insights, where the hosting Operating System, the running Device, and the network interface were the main culprits of the system's energy consumption for whatever configuration.

4.2.1.2 Comparative analysis

Then, we evaluated and compared the two implementations of our approach (i.e., DDbS and SRS). To prove that SAVRUS results are correct, we required completely measured variability models for a well-known quality attributes. Consequently, we opted to test the accuracy of SAVRUS for quality attributes atruntime as the results are aligned with the quality attribute energy consumption. The variability models, detailed in Table 14, are: Dune, HSMGP, HiPAcc [48] and Trimesh [49]. To emulate our issues, we purposely removed random chunks of measurements mimicking the issues of large variability models modelling energy-aware VNFs systems (i.e., randomly spread measurements). To obtain domain unknown configuration spaces with scattered measures, we degrade the data by randomly erasing parts of the variability models measured space. For scalability testing, we increased the number of samples from 25 to 6,400.

We validated SAVRUS effectiveness regarding: (i) the quality of the features insights; (ii) the size of the sample sets with respect to the space size; and, (iii) the analyses times. SAVRUS generated a correct ranking of noteworthy features and interactions 80% of the times for every incompletely quality attribute measured model. Regarding SAVRUS performance, the current prototype has a base runtime of 1 minute, taking less than 3 minutes for comprehensible cases and scenarios. Between the two sampling implementations, DDbS was the most balanced alternative due to its speed, accuracy, and scalability, especially for large and complex systems. Anyhow, if the analysis time is not an issue for a developer, SRS is the most accurate alternative.

4.2.1.3 Conclusions and future work

Currently, we have developed a SAVRUS prototype and proved its usefulness with a large case study from the domain of Edge computing systems, comprising 554 features and a total of 10^8 legal configurations. This case study didn´t consider the variability of VNFs. We plan to apply our approach to reason about the energy consumption of highly variable VNFs required in the context of Augmented Reality applications. Specifically, with an offline SAVRUS analysis, we will create a learned energy model

[10] https://github.com/gsdlab/clafer.

that transfers the knowledge to real-time algorithms acting on predictors or heuristics. If the network, functions, services or devices are updated, the energy model will evolve with an extra offline adjustment executing SAVRUS again. We expect to improve the speed and accuracy of real-time and dynamic algorithms in the second iteration of the project. This solution will impact mainly KPI K1, and its results will be presented in the next deliverable of WP5.

4.2.2 Combining VNFs at the edge (A7)

In this Section, we present the results of the evaluation of the algorithms present in Section 5.3.3 of deliverable D3.1 [34]. These two algorithms are the Application Variability Adaptor (AVA) and the New Devices Finder (NDF); they target the configuration of VNF requirements and edge-based infrastructure capabilities for a better the estimation of energy consumption.

4.2.2.1 Campus-wide cyber physical system use case

By now, we have applied the AVA and NDF to a real case of an academic campus where several devices, those typical of cyber physical systems (CPSs), are geographically distributed serving different applications. The campus infrastructure includes sensing units, IoT gateways, computers, cloudlets, and dedicated cloud servers, scattered across the campus. These devices are not using all their computation and communication capacities (or even are suspended most of the time). All of them are located at the far edge of the Internet, connected to the campus institutional access network. Both algorithms were able to obtain an optimal solution for their given problems in the case study.

The time needed by our modules to provide a solution varies according to the size of the problem [50]. For this reason, we have evaluated the execution time for different problem sizes. With this purpose, we have developed a Benchmark version of the algorithms, which allows setting the number of features and devices in case of the AVA and the number of services and devices for the NDF. Each experiment is performed 30 times on one thread of an AMD Ryzen 7 1700X processor.

In all the experiments the infrastructure considered is formed by 30 different devices with arbitrary characteristics (set on each experiment). These random characteristics involve software and hardware components, as well as the location. Table 15 shows the results.

Table 15. *AVA and NDF algorithms execution time.*

Problem size: AVA: (D/F) NDF: (D/S)		**30/10**	**30/20**	**30/30**	**30/40**	**30/50**	**30/60**	**30/70**	**30/80**	**30/90**	**30/100**
AVA	Mean (s)	0.07	0.14	0.24	0.41	0.60	0.88	1.39	1.99	2.95	4.23
	Std	0.01	0.01	0.01	0.02	0.02	0.03	0.03	0.03	0.04	0.03
NDF	Mean (s)	0.12	0.50	1.09	1.97	3.03	4.49	6.03	7.94	10.11	12.21
	Std	0.00	0.01	0.01	0.02	0.03	0.07	0.06	0.06	0.16	0.30
D: Devices. F: Features. S: Services.											

4.2.2.2 Algorithm performance

For the first algorithm (AVA) the number of features has been incremented up to 100. Experiments show that the AVA returns a solution almost instantaneously in most cases, requiring around 4.2 seconds in the worst case of a very big application formed by 100 heterogeneous features.

Regarding the experiments of the NDF, the number of services has been incremented from 10 to 100. Services characteristics such as computational load, data to transmit, location requirements, type of device, peripherals, sensing units, operating system and interaction ways have been randomly set for each experiment. Results show that, for an application formed by 10 different services, the NDF needs around 0.12 seconds to return a solution, being 12.2 seconds in the worst case of an application formed by 100 different services.

4.2.2.3 Conclusions

The utility of the AVA and NDF is shown by applying them to manage the evolution of an application and the infrastructure of a real IoT scenario [51]. The execution time of our algorithms for different problem sizes has been also evaluated using a *Benchmark*, and the conclusion is that they return a solution in a reasonable amount of time. These solutions will have an impact on both KPIs K1 and K2.

4.2.3 AI-enhanced MANO (A8)

This activity targets the development and experimental assessment of a platform for AI-supported MANO. Currently, the work has focused on the integration of the components of the platform towards deploying a comprehensive and flexible architecture that will enable structured MANO tests in presence of NI.

4.2.3.1 *Platform components*

The AI-enhanced MANO solution is deployed on a set of servers on which the OSM and OpenStack are deployed. To enable the intelligence at the orchestration and management level, multiple interfaces with different components are used. Specifically, the connection with Open-Source MANO has been enabled through the OSM's Northbound API featuring ETSI NFV SOL005. Communication with OpenStack has been done through its provided API with a RESTful web service endpoint to access, provision, allocate and automate the resources. Data and logs are retrieved directly from Elasticsearch REST API, that is also used in cases of configuration and access to other Elasticsearch features. Additionally, custom APIs have been designed and developed for enabling the acquisition of information from VNFs, applications and infrastructure with the use of exporters that send specific metrics to the Elasticsearch and Diagnostic components. Finally, the connection with Verticals has been performed with a custom API that directly connects the AI-enhanced MANO with the Verticals retrieving in this way all requirements or specific SLAs.

4.2.3.2 *Plartform architectural design*

The architecture of the AI-component is shown in Figure 35 and its connections and functions are explained below.

1. Direct Verticals info (i.e., Metrics/KPIs) of the Network Service (NS) that will be deployed acquired from AI-enhanced MANO or by utilizing the 5G EVE APIs.
2. Internally an API SERVER gets the request and triggers the AI component to evaluate the optimal deployment of the VNFs based on Vertical requirements and KPIs.
3. The API SERVER gets the VNF's info from the OSM (OSM NB API) and historical data from db.
4. The API SERVER retrieves the metrics exported from the MEC platform and the MEC's VNF instances, to evaluate the metrics in real time.
5. The AI component acquires all information in real time (MEC's metrics, VNF IDs, and VNF requirements) from the API SERVER and uses old data and previous decisions. The AI finds the optimal deployments and migrations for the NSs in MECs.
6. The AI component triggers the OSM (OSM NB API), through API SERVER, for any migration or reallocation of resources.

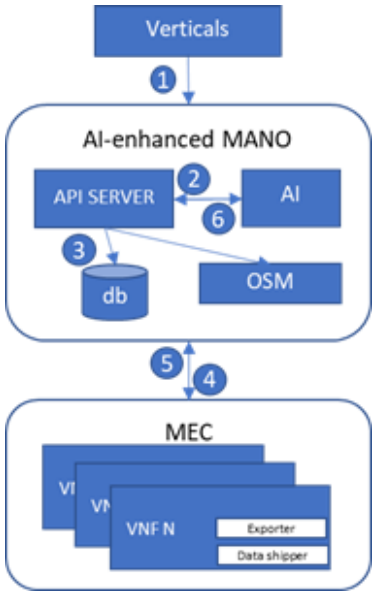


Figure 35. *AI-enhanced MANO solutions.*

4.2.3.3 *Summary and future work*

With the platform deployed, the activity will focus on integrating NI instances for MANO therein. Thus, in the second iteration of the project, we will work towards designing and plugging into the AI block in Figure 35 smart algorithms for improved MANO. The results of such integration, in terms of engineering work and possibly of performance evaluation results, will be reported in the next deliverable of WP5.

4.3 NI for real-time anomaly detection

Evaluation E3 focuses on NI solutions that support anomaly detection in real-time in both controlled environments and in a production core network. The DAEMON consortium performed assessments of challenges and solutions related to E3 via activity A9. **Table 16** summarizes the tools, KPIs, TRL, PoC plans, approximate progress and main innovations of such activities.

Table 16. *List of activities for E3.*

ID	Name	Evaluation	Tool	Planned KPIs	Collected KPIs	Target TRL	Planned for PoC demo	Progress
A9	Federated Learning-based Anomaly Detection	**E3**	T8	K3, K7	K3	4	Yes	40%
	Main innovation: Use federated learning for improving the efficiency of Anomaly Detection							

A single activity A9 is targeting E3, by investigating Federated Learning-based anomaly detection. Details on A9 are provided next. Note that the consortium is considering the option of merging this evaluation with E5, given the proximity of the two evaluations and the fact that only one activity was carried out in E3 and two in E5 to date. We will assess the advantages of this option, and possibly consolidate E3 and E4 during the second iteration of the project.

4.3.1 Federated Learning-based Anomaly Detection (A9)

B5G/6G infrastructures 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, from video streams, or even audio; these streams can vary in requirements, for instance in terms delay and reliability. All these aspects (demands and system organization) can appear organized in the form of a set of slices that are concurrently supported by the infrastructure. Bypassing the aspects of whether slicing is used, and by adopting a more general perspective, it can be assumed that the network will be asked to serve a sheer amount of traffic sources. These traffic sources should exhibit a behavior within a certain framework. In parallel, the system should provide a certain (agreed) service to each traffic source, and to the aggregation of sources, per geographical area and time zone.

4.3.1.1 Anomaly detection in presence of shared sources

However, things do not always operate as they should, or as agreed. Therefore, systems should be prepared to handle situations that exhibit a behavior beyond what is "ordinary", agreed, etc. 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. In other words, anomalies can be security incidents or may indicate faulty sensors, or may be related to aspects of interest to the vertical domain (i.e., the concern is on the data and on the control plane).

In light of the above, a key problem that needs to be solved can be generally stated as follows: "Given (a) an area / network segment, (b) the verticals / services supported in the area and their anticipated / agreed behavior in time and space, and (c) the network configuration set up for supporting the services, find the sources that exhibit and unordinary behavior".

The problem above can fall in the class of problems that is generally called "anomaly detection". The problem has attracted attention in various domains and for various solution approaches. Our main approach will be to rely on unsupervised learning and pattern recognition. This are representative solutions that enable the experimentation with diverse levels of data availability.

4.3.1.2 An architecture of FL anomaly detection

Our architecture will follow the Federated Learning (FL) approach [52], for reasons of scalability and data protection. Scalability is important in our case since we consider an environment with numerous traffic sources. Likewise, the aspect of keeping data locally is important. Therefore, our work will experiment with the FL approach for the realization of the anomaly detection mechanisms.

Figure 36 gives an overview of our approach. In high level terms, the inputs, outputs and algorithm are indicated. A knowledge base is highlighted. It will contain local information regarding the performance of the model. Certain information is communicated to a controller of the FL model. Likewise, the controller will be tuning the algorithm in accordance with the FL paradigm.

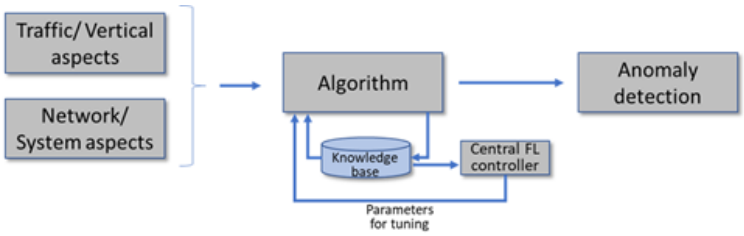


Figure 36. *FL-based architecture for anomaly detection.*

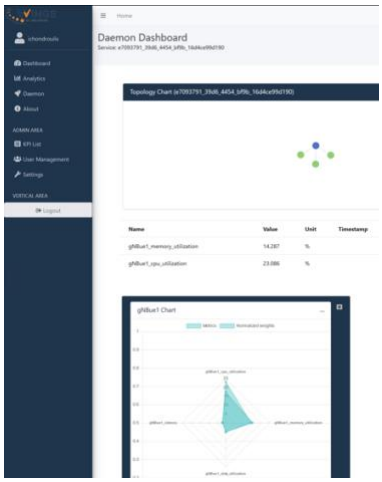


Figure 37. *Dashboard.*

4.3.1.3 *Summary and future steps*

The activity has currently focused on the implementation of the FL-based architecture, including the coding of a graphic user interface, a sample of which is shown in Figure 37. Therefore, no experiment has been run to date. Based on the scoping so far, the next steps concern the development of models, the realization of validation activities (result collection and analysis), and the support of dissemination and demonstration activities. These will be carried out during the second iteration of the project and the results will be presented in the next deliverable of WP5.

4.4 NI for Edge orchestration

Evaluation E4 targets NI solutions for service orchestration and resource allocation algorithms in the Edge micro-domain. The DAEMON consortium performed assessments of challenges and solutions related to E4 via activity A10-A17. **Table 17** summarizes the tools, KPIs, TRL, PoC plans, approximate progress and main innovations of such activities.

Table 17. *List of activities for E4.*

ID	Name	Evaluation	Tool	Planned KPIs	Collected KPIs	Target TRL	Planned for PoC demo	Progress
A10	Video analytics with edge computing	**E4**	D5	K2	K2	3	TBD	70%
	Main innovation: Compute-aware scheduling for analytics VNF							
A11	Multi-timescale edge orchestration	**E4**	T4	K2, K5, K8	K8	4	Yes	50%
	Main innovation: Multi-timescale management and orchestration of edge services and resources							
A12	WLAN performance prediction for spectrum management	**E4**	D6	K5, K8	None	3	No	10%
	Main innovation: Enabling fast WLAN performance prediction for spectrum optimization							
A13	Data-driven resource orchestration	**E4**	D1	K5, K8, K9	K5, K8, K9	4	No	30%
	Main innovation: Clustering of device population to extract resource requirement profiles							

A14	Multi-timescale network slice reservation	**E4**	None[11]	K4	K4	2	No	60%
	Main innovation: Slice reservation policies for optimal resource employment based on OCO							
A15	Testing EnergyEdgeCloudSim	**E4**	S4	K1, K2, K3, K4	K1, K2	3	TBD	10%
	Main innovation: Enery-aware VNF orchestration							
A16	Towards autonomous VNF scaling	**E4**	S1	K4	K4	3	No	50%
	Main innovation: General-purpose NI solution for autonomous VNF auto-scaling at edge							
A17	Auto scaling Virtualized RAN caches	**E4**	D9	K3, K4	K3, K4	2	No	30%
	Main innovation: Jointly optimize dynamic cache rental, content placement, and request-cache association in wireless scenarios							

Overall, the preliminary results of these activities already led to a number of key observations on the orchestration of services and allocation of resources in the Edge micro-domain, as follows.

- ***A first line of activities target an improved, flexible and automated management & orchestration of Edge resources based on NI solutions***. Here, the proejct activities ***A10***, ***A11***, ***A15***, ***A16*** and ***A17*** have demonstrated NI algorithms that span across multiple aspects of the problem. Specifically, these aspects include: (i) how to dynamically instantiate and auto-scale Edge instances and the VNFs they run, so as to best serve mobile devices and reduce the energy footprint of the mobile Edge; (ii) how to best offload computational tasks from mobile devices to such deployed Edge instances; and, (iii) how to cache contents at the Edge instances so as to maximize the QoE.
- ***A second line of activities for this evaluation aimed at investigating NI solutions in sliced Edge environments, by studying the impact of NI solutions on the management of specific mobile services***. More precisely, the project carried out activities ***A13*** and ***A14*** that studied (i) algorithms for managing pricing of the service provider market in network slicing settings at the Edge, and (ii) a characterization of the current 5G deployments towards supporting service-specific M-IoT (i..e, multimedia IoT) requirements.
- ***Finally, we carried out comprehensive comparative assessments of different types of NI models, including based on statistical, control and machine learning tools, for Edge orchestration***. These activities were performed in ***A12*** and ***A16***, and showed that there is no one-size-fits-all approach, and the most appropriate model must be picked based on the specific Edge orchestration task. For instance, our results prove that a Graph Neural Network (GNN) grants substantial gains in performance with respect to simpler models for Edge prediction problems; however, Deep Q Learning (DQL) incurs into substantial operational costs that make simpler threshold-based solutions preferable for Edge instance autoscaling.

4.4.1 Video analytics with edge computing (A10)

This activity assesses the performance of solutions for the offloading of tasks from mobile devices to the mobile edge, using algorithms presented in Section 4.4 of deliverable D4.1 [46]. The goal is improving end users' Quality of Experience (QoE) by taking scheduling decisions (i.e., dynamically adapting the offloading parameters) in an efficient manner.

4.4.1.1 Video frame recognition at the mobile edge

We consider object recognition in video frames as the analytics task, where a mobile device capturing a video aims to recognize the object in those videos with the maximum accuracy possible and minimum latency. To this end, mobile devices can offload computationl expensive operations to an edge server. To this end, the NI must decide on (i) how to set the compression ratio for each device, (ii) the time slice dedicated for processing the device data at the edge facility, and (iii) the input size to the neural network deployed on the edge facility to process such data. Details are in Section 4.4 of Deliverable D4.1 [46].

[11] At this level of the development, activity A14 has been validated by using simple Python scripts.

4.4.1.2 Overall performance

First, we consider two users with minimum frame-rate requirements of λ = 10 and 20, and we plot the average regret with time in Figure 38 (left), together with a light blue shaded area indicating the 1-std area over 100 evaluations. The figure confirms that the policy (i.e., online decisions) is zero-regret. In Figure 38 (right), we see the performance in terms of cumulative confidence and frame rate simultaneously (on different axis). Initially, the images are over-compressed, leading to a smaller data size and lower latency. Thus, we observe the high frame-rate, but an unsatisfactory performance in terms of cumulative confidence. However, as time progresses the policy is more balanced and does achieve high confidence while respecting the required frame-rates.

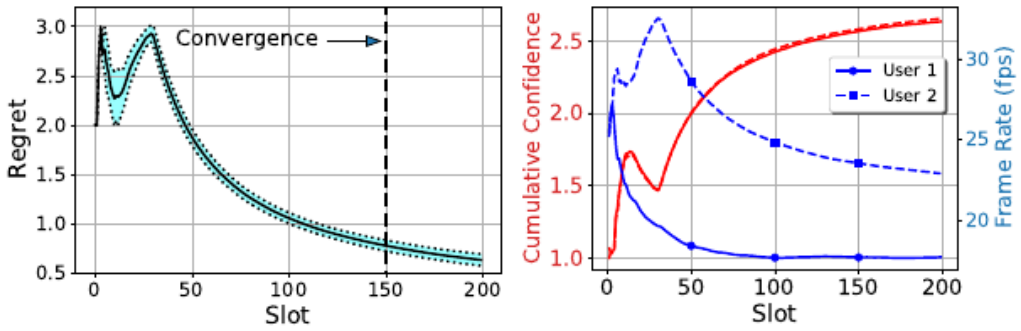


Figure 38. *(Left) Average regret of the proposed method. (Right) Cumulative confidence & frame rate.*

To evaluate the scalability of the approach presented in Section 4.4 of deliverable D4.1 [46], we measure its average iteration delay. Figure 39 (left)depicts this delay as a fraction of slot duration Δ for different number of users N. We observe that for the first 30 slots (expansion stage), the delay increases both with the slot t (matrix inversions of size t) and users N (more configurations), but when safe set has been fixed and posteriors do not require updates, it drops substantially. After that, the iteration delay starts increasing again with t for the same reason as before, but is kept low until the algorithm converges to an acceptable solution. We consider more users in Figure 39 (right) where we set a low frame rate requirement λ = 2. In the top graph, we show the maximum value of the iteration delay within a 200 slots evaluation and in the lower graph, we show the slot in which (on average) the stopping of our algorithm occurs for different values of N. For the former, we can see that the delay gets much bigger than the slot duration for $N \geq 12$ and for the later, that the differences are insignificant and that we can always stop the algorithm in fewer than 200 slots.

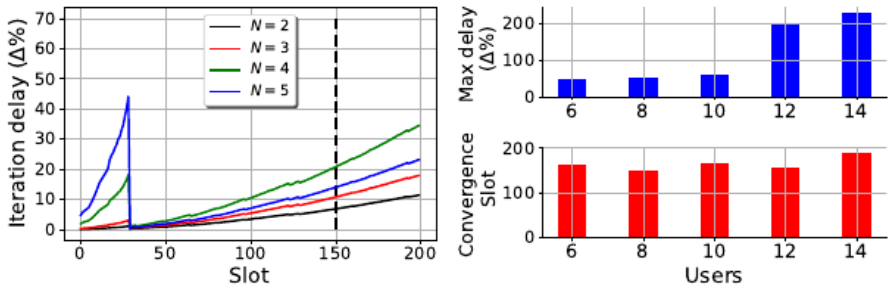


Figure 39. *(Left) Algorithm mean iteration delay. (Right) Maximum iteration delay & convergence time.*

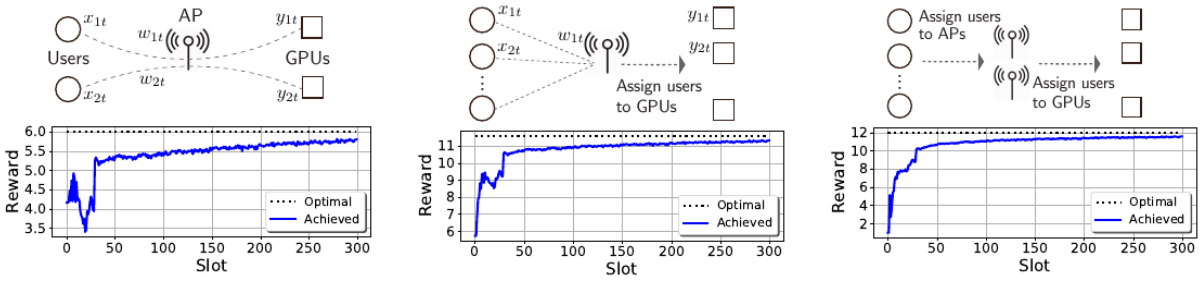


Figure 40. *Reward of (a) preassigned users, (b) user-to-GPU assignment, (c) AP-to-GPU assignment.*

4.4.1.3 Generalization to additional settings

To demonstrate the generality of the proposed solution framework, we additionally evaluate the framework for three different settings: (i) Multiple GPUs (i.e., each user can configure the neural network size differently), (ii) the number of users N is higher than the number of GPUs, and (iii) the users can be served J Access points. Specifically, we set $N = 4; K = 2; J = 1$ for Scenario (ii), and $N = 4; K = 2; J = 2$ for Scenario (iii). From Figure 40, it can be seen that the reward (sum of confidences) keeps improving towards the optimal one. The observed performance is within 6%, 4% and 5% of the optimal in each Scenario, after 200 slots.

4.4.1.4 Conclusions

We developed and evaluated a compute-aware online framework that allows configuring mobile edge resources and models so as to best accommodate offloaded operations from mobile devices, and maximize the end users' QoE. The results show how the proposed approach can firstly identify feasible networking and learning parameters, and then deduce online actions that provably reach those that are the best ones in hindsight.

4.4.2 Multi-timescale edge orchestration (A11)

To cope with the challenges of manual NFV MANO operations, which are a reactive approach that causes delayed operations, there is a need to switch to proactive one that is characterized by automation and intelligence in operations of orchestrating services and resources. The need for automated and more optimized orchestrations becomes critical for the services with stringent requirements for latency and capacity, such as those that belong to vehicular system. Thus, in this activity, we utilized the real-life testbeds, Smart Highway (T4) and Virtual Wall, to create a PoC for pursuing realistic experimentation and validation of the impact that AI/ML models have on the edge orchestration.

4.4.2.1 Collecting and analyzing training data

To collect training data (response time measured at client side), we created a scenario in which we utilized the PoC described in Section 3.1.4, and we gradually stressed the edge V2X deployment on the RSU 3 by using stress tool Locust. If we observe the average response time presented in Figure 41, we can see how much communication and computational delays are contributing to the overall edge service response time. Samples indicate 20 batches of successive measurements, where each of the measurements lasted for 5 minutes, and is represented by the mean value. The stress test in our scenario caused an increase in average response time, and as we can see in Figure 41, communication latency remains stable despite the stress test, thus, the computational latency on the edge node is affected.

In Figure 41 (b) we show the average values of CPU load, RAM load, and power consumption, in the Kubernetes cluster on the Edge node 1, i.e., RSU 3. Samples of measurements correspond to the samples of edge service response time in Figure 41 (a). Given that the scenario indicates a gradual stress test from sample 1 to sample 20, in Figure 41 (b) we can see an increasing trend in CPU load, and the goal is to explore the dependency of service quality experienced by users (i.e., vehicle) on infrastructure metrics, such as CPU load. We further exploit this dependency to improve the service quality experienced by user (i.e., vehicle), while other collected metrics are used by the MCDM algorithm to improve the final decision on service relocation (e.g., avoiding using an edge node with high power consumption). In this experimentation setup, we used python to apply two types of Support Vector Regression (SVR) depending on the kernel, i.e., Radial Basis Function (RBF) and Linear. Finally, we created two datasets, one for training (gradually applying stress test), and another for testing (randomly applying stress test).

4.4.2.2 Performance results and discussion

Figure 42 (a) shows the prediction of an average response time based on the training data, while Figure 42 (b) presents the prediction based on the testing data. As we notice that SVR with RBF kernel produces larger R-squared value (better fits the input to output), and lower Mean-Squared Error (MSE) (determines the accuracy of our model), this model is further used and applied in our algorithm for selecting the edge deployment.

The SVR model achieves a high value of R-squared, i.e., 0.9979 (cf. Table 18), and produces an MSE of 2.64471, which can be considered as a satisfactory level of prediction accuracy, given that average difference between predicted and measured data is less than 1ms (0.6651ms) with standard deviation of 1.484ms. For the type of V2X use cases where notifications/warnings are generated and collected from edge services, to extend the contextual perception of a vehicle, the result we obtained can be considered as satisfactory due to the following reasons. In case a vehicle is moving with an average speed of 80km/h, 15ms can be considered as a tolerable latency for retrieving important warnings, as a vehicle moves only for 0.33m until it gets a new notification. This of course needs to be studied with a more prominent attention in case of autonomous driving, or teleoperation of a vehicle. Finally, Figure 42 (c) shows the result of the gain in average service response time that can be achieved by performing edge V2X service relocation in a proactive and automated way. As the cloud orchestrator is constantly monitoring CPU data from different edge domains, it applies SVR model to predict the average response time for a particular type of edge V2X application. If predicted values of average response time in the

upcoming sample (lower than 5min) are larger than the threshold, which we consider as 15ms for a used type of service, then the cloud orchestrator applies the MCDM, and potentially requests an application relocation to Edge 2 (RSU 5) from Edge 1 (RSU 3). In Figure 42 (c), this happens in the sample 17, where the decision for application relocation is applied by edge orchestrators, and vehicle client starts consuming edge V2X service deployed on the RSU 5.

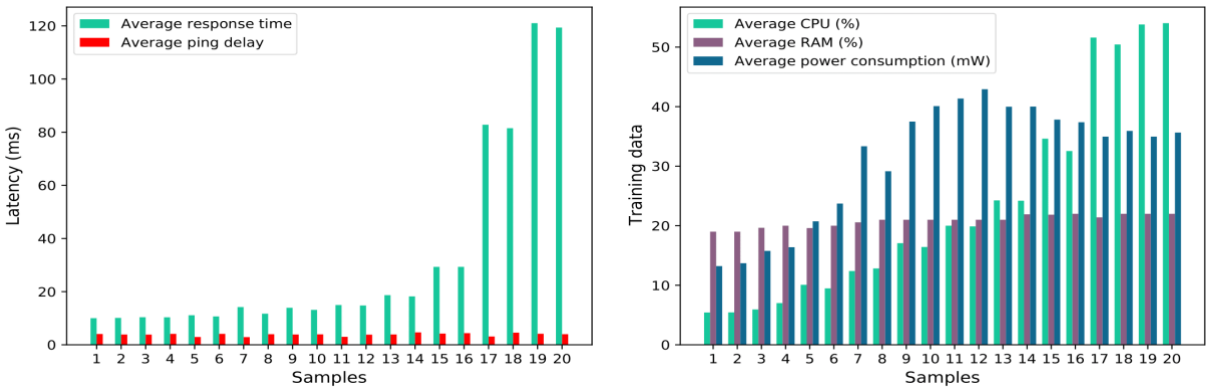


Figure 41. *(a) Latency, (b) Training data*

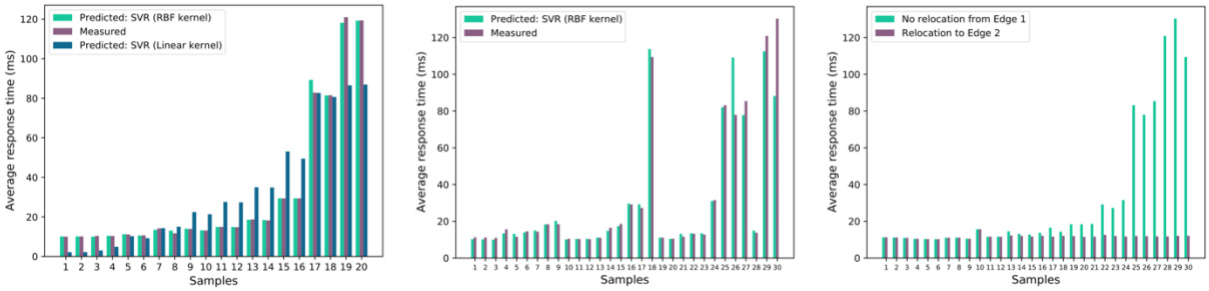


Figure 42. *Results: (a) Prediction based on training data, (b) Prediction based on testing data, (c) Gain achieved by service relocation.*

We can clearly see the benefit of applying proactive relocation and re-attaching the user from one edge to another, as from sample 17 onward, clients would experience an increased average response time if no relocation happens (no relocation from Edge 1). Hence, our model improves NFV MANO operation making a quality-aware decision to proactively instantiate instance in the target edge domain, and to re-attach vehicle client from one edge to another. This is particularly visible in sample 29, in which the average response time decreases for 92.3% if the relocation happened. On the other hand, a simple rule-based algorithm that compares a single predicted value of latency with a predefined threshold might be inefficient, as they might lead to frequent requests for service relocation that need to be handled by edge orchestrators.

Table 18*. Results of multi-timescale edge orchestration (Average is an average difference between measured and predicted data).*

Model	R-squared	Mean Squared Error (MSE)	Average	Standard deviation
SVR (RBF Kernel)	0.9979	2.64471	0.6651ms	1.484ms
SVR (Linear Kernel)	0.8277	221.8706	9.985ms	11.7372ms

4.4.2.3 *Conclusions*

We utilized the real-life testbeds, Smart Highway (T4) and Virtual Wall, to create a PoC for pursuing realistic experimentation and validation of the impact that AI/ML models have on the edge orchestration. We presented our progress on improving MANO operation of service relocation towards achieving service continuity and required service quality, by applying an ML-based quality-aware concept that automates service relocation and minimizes average vertical service response time. In the second iteration of the project, we will work towards refining the NI design and performing more complete assessment of their quality in our target real-world scenario. Results will be presented in the following WP5 deliverable.

4.4.3 WLAN performance prediction for spectrum management (A12)

In this activity, using the dataset D6, described in Section 3.3.6, we trained and evaluated four state-of-the-art ML models to predict the throughput of all devices in an enterprise-like scenario containing a different number of APs and STAs applying DCB. The ML models were described in deliverable D3.1 [34]. Two deep learning based models, a CNN and an FNN, and a Gradient Boost (GB)-based model were designed to predict the throughput in IEEE 802.11 WLANs that support DCB. To complement the models, a Graph Neural Network (GNN) architecture was also designed. Details are provided next.

4.4.3.1 ML models comparison

As a baseline for all ML approaches, we use a random guesser. We assume that the throughput can be obtained from a normal distribution with the mean and standard deviation found during data analysis to build this random guesser. Given that the throughput in STAs varies between 0 and 88 Mbps (cf. Table 7), we use a truncated normal distribution between those values. This random approach represents a naive and cheap way to generate predictions in this particular problem.

The models were trained on the corresponding data set with a fixed split (80% for training and 20% for validation). Every model uses the Root Mean-Squared Error (RMSE) as a loss function. The error was obtained across the predictions ($\hat{x}_i$) compared to the actual results (x_i), where N is the number of devices in the batch. We trained the models using different combinations of all available features (cf. Table 19) to quantify how they affect the prediction accuracy. This procedure was performed ten times per experiment to observe each model's convergence.

Table 19. *Features available for training.*

Feature	Definition	Feature	Definition
Node Type	Wireless node type, AP = 0, STA = 1	**Distance**	Euclidean distance between AP and STAs
x(m)	x-coordinate of the wireless node	**Bandwidth**	Maximum channel bandwidth
y(m)	y-coordinate of the wireless node	**RSSI**	Received Signal Strength Indicator
Channel Configuration	Combination of Primary, minimum and maximum channel	**Interference**	Inter-AP interference sensed from every AP (mean)
SINR	Signal to Interference plus Noise Ratio	**Airtime**	Percentage of time each AP occupies each of the assigned channels (mean)

The trained models were used to predict the throughput of all devices in the test data set. Figure 43 shows the mean RMSE across all test scenarios' deployments in a given experiment for all the generated models and its standard deviation. As can be seen from the figure, GNN outperforms all other approaches in all defined experiments. The random approach performs the same, independent of the features used. However, learning from data represents at least a 20% improvement regarding this random approach, using all trainable features (Exp 1). Focused on Exp 1, i.e., the experiment with all features, GNN can obtain up to 64%, 56%, 55%, and 54% when comparing it against the random approach, the CNN, the FNN, and the GB, respectively. Surprisingly, GB performs slightly better in several experiments when compared to the CNN and the FNN. Despite its complexity, the CNN does not perform better than the FNN and the GB. This poor performance might be due to data representation. CNNs outperform other ML approaches when dealing with high-dimensional data (e.g., images, time-series). Even though the wireless environment is too complex to be modeled, the provided data do not include an extra dimension (e.g., time) that CNN can benefit from.

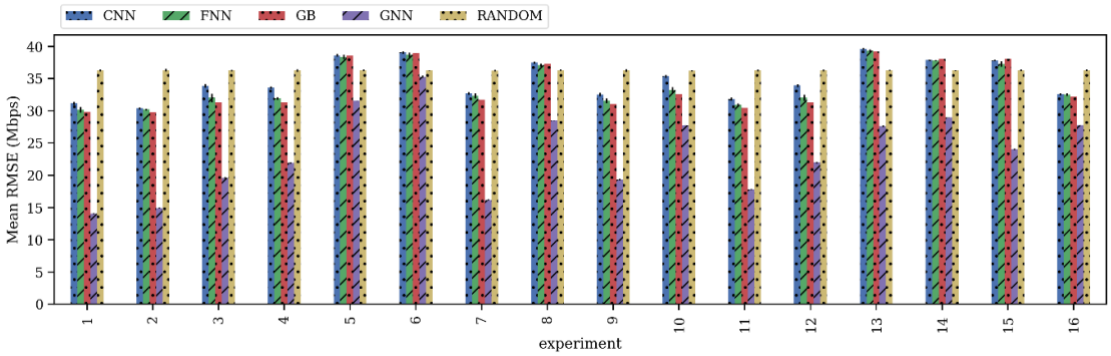


Figure 43. *Mean and standard deviation of the obtained RMSE by all models on the test data set.*

In some experiments (e.g., Exp 5, Exp 6, Exp 8, Exp 13, Exp 14, and Exp 15), the random guesser performs better than some ML approaches since the combination of input features does not benefit from the learning process. In fact, the Gaussian assumption works well in some cases, and it is widely adopted in many communications aspects. Moreover, the random guesser performs the same, independent of the input features, while for some experiments, the ML approaches benefit from certain features (e.g., Exp 1, Exp 2). This random guesser includes some basic information and performs better than a best-effort approach. It serves as an upper bound to the ML approaches, as it shows if the ML model is learning.

4.4.3.2 Feature relevance

In terms of feature relevance, including airtime, there is a strong improvement to the results of all ML models. For instance, the only difference between Exp 1/Exp 7 and Exp 15/Exp 8 is that the latter does not consider airtime while the former does. It can be seen that not considering airtime represents between 85% and 87% decrease for the GNN, while other ML approaches perform even worse than the random approach. Nonetheless, considering only airtime as an input feature does not ensure good performance. For example, in Exp 16, GNN decreases its performance by more than 100% regarding Exp 1, except for other ML approaches, where using airtime as an input feature performs even better than considering the rest of the features (see Exp 15).

Analyzing other features, RSSI and distance give more information about the throughput than SINR, node type, and interference. For instance, in Exp 9 and Exp 10, all models improve their performance by including RSSI: 48%, 5%, 10%, and 6% in the GNN, GB, CNN, and FNN, respectively. Similarly, in Exp 11 and Exp 12, GNN obtains around 26% improvement, while CNN and GB obtain 2.4% and 7.5% improvement, respectively, when considering the distance. Interestingly, the GNN is the only model in which features such as node type (Exp 3 vs. Exp 4) and SINR (Exp 5 vs. Exp 6) are relevant and improve its performance. Even when considering interference (Exp 13 vs. Exp 14), a factor that seems to decrease other ML models' performance, GNN obtains a 5% improvement.

4.4.3.3 Conclusions

Network models and optimization algorithms are developed to offer a high degree of automation to accelerate service delivery while meeting economical goals. By learning from data, Neuronal Networks are able to build a function that abstract complex network behavior. Our comprehensive comparative evaluation shows that GNN models are especially well suited to the WLAN performance prediction problem, where information is also embedded in the topological representation.

4.4.4 Data driven resource orchestration in the MNO (A13)

This activity targets the development of novel algorithms and solutions based on new concepts of cellular networks to support multimedia IoT (M-IoT) application requirements. As a starting point, to recognize the demands of these applications, we have studied Connected Cars as a use case in this deliverable. Then, we will focus on devising learning-based resource management techniques to meet these requirements. Moreover, learning algorithms can be leveraged to predict near-future traffic patterns, and such predictions enable proactive and accurate resource management decisions.

Relying on dataset D1, we explore the performance of the operational 5G network at the radio sector level and capture the demand from the population of devices that the operator serves. With this, we then work towards identifying fine-granular clusters of devices with similar requirements in terms of quality of performance and network resources. Having these profiles is an important input for achieving efficient resource orchestration.

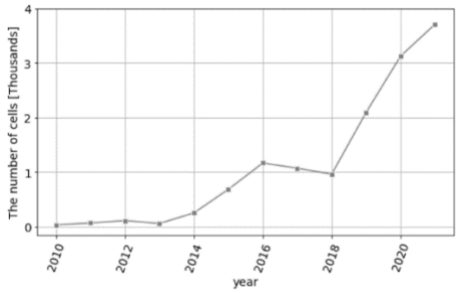


Figure 44. *Yearly trend of cell sites launch over the past decade.*

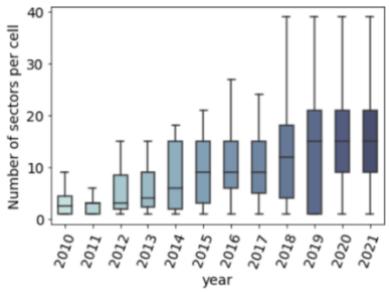


Figure 45. *The number of radio sectors per cell site over the last decade.*

4.4.4.1 Network deployment evolution

We start with an analysis of the radio access network deployment evolution. Monitoring this information together with the performance status of the network allows us to capture from the real-world dataset the operator's decisions in terms of deploying new cell sites and configuring new carriers at these sites. The results we present in this section open the door for exploring network intelligence approaches for

automatic and accurate generation of new carrier configuration parameter values using learning and recommendation techniques. We first investigate the network deployment in terms of the new number of cell sites that are deployed and activated for different technologies. In Figure 44 and Figure 45**Error! Reference source not found.**, we illustrate the number of cell sites and sectors per cell deployed annually over the last decade. We note that while over 95% of the cell sites in recent years are 4G, they also are equipped with 2G/3G. We see especially a sharp increase in 2019, which we believe is a response to the substantial growth in the number of cellular devices (both new cellular IoT devices, and smartphones).

We turn our attention to 5G deployment and have a closer look into the network deployment during the last two years, since commercial 5G support from the MNO started. In our dataset, we record radio sectors that had radio signalling activities. We count the number of the active sectors on the first date of every month from January 2020 to November 2021. The motivation behind counting the number of sectors instead of cell sites is to show and accurate evolution in the available resources in the network.

Due to the operator confidentiality, we only present the delta variation over the first day of January 2020 in Figure 46. We observe that 5G capable devices started to connect to the 5G sectors in December 2020. Furthermore, over the last two years, the number of 5G active sectors has increased 90.3%, and continues to present an overrising trend.

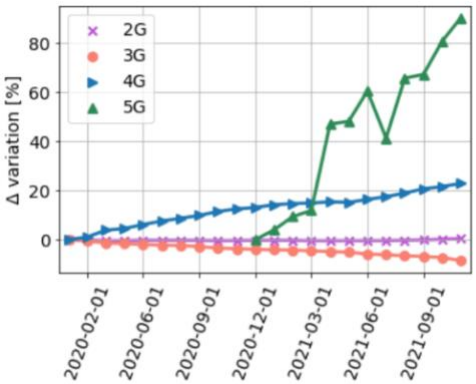


Figure 46. *Delta variation of the number of active sectors per radio access technology. We employ as reference the first measurement date on the x-axis.*

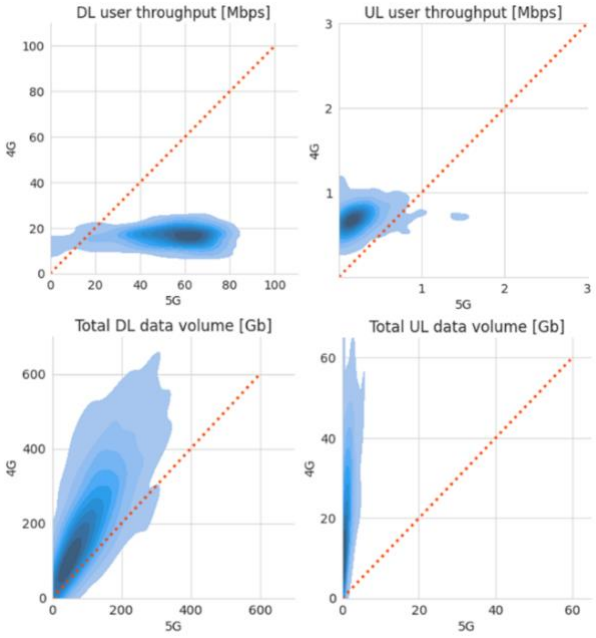


Figure 47. *Daily performance of 5G vs. 4G sectors in the same locations (for a non-standalone 5G deployment). Each plot indicates the median of the metric we show in the title per geolocation.*

4.4.4.2 5G network performance

We investigate the network performance metrics over all bearers corresponding to QoS class Identiters (QCIs) of 5G sectors as compared to 4G sectors through two months (September-October) in 2021. By taking median values over two months, we minimize the impact of abnormal traffic behaviour, special events, or the differences between days of the week. We aggregate per sector hourly metrics per day and per location and compare median of each performance metric of 4G and 5G sectors in the same location. We begin our analysis with throughput performance (average DL/UL throughput over all users).

We show in Figure 47 that in 75% of locations 5G technology can increase the median of DL user throughput at least 157% (i.e., median of 5G and 4G are 59Mbps and 17Mbps, respectively). While we have been failed to see any 4G sectors with DL user throughput more than 34Mbps, more than 75% of 5G sectors gain at least DL user throughput of 36Mbps. We also compare it with the DL user Carrier-aggregation (CA) throughput of 4G sectors and find on average 203% increase. On the other hand, comparing UL user throughput of 5G and 4G sectors indicates only in 7% of locations 5G sectors provide more UL user throughput. Plus, we observe a 90% drop when comparing median UL user throughput of 5G with 4G in the 25% of the locations.

However, considering small UL data volume, throughput may not be a good measurement to evaluate performance. This is because most of the traffic may only use a single TTI, and may be excluded from the data volume counters. We note total amount of DL data volume per5Gsector is still 50% lower than 4G

sectors. Meanwhile, the total number of connected 5G users, and DL active users are 97% and 72% less than 4G (comparing their median).

4.4.4.3 *Conclusions*

In this section we explored the performance of an operational cellular network, relying on dataset D1. We first analyzed the evolution of 5G sites, showing that an increase of 90.3% in the number of sites have been achieved. Second, we analyzed the 5G network performance in terms of throughput. The most interesting results are a 157% increase in DL with respect to 4G and the total amount of 5G DL data volume 50% lower than 4G while the number of DL active users is just 72% lower than 4G. These results suggests network traffic growth by adoption of 5G in the following years. In the second iteraton of the project, we will build on these insights to build a data-driven approach to develop, implement and experiment with radio resource management algorithms.

4.4.5 Multi-timescale network slice reservation (A14)

In this activity, we carry out simulation experiments to evaluate the performance of the Online Learning for Reservations (OLR) algorithm and its extensions, proposed in Section 5.5.1 of Deliverable 3.1 [34], in different scenarios. Specifically, a slicing market with the following properties is considered: the Network Operator (NO) manages B cellular base stations connected through a backhaul network of 100 paths to N = 20 core nodes with data processing and storage capabilities. Thus, the Service Provider (SP) reserves slices with the following four resources: wireless and backhaul bandwidth, storage, and CPU capacity. The maximum slice size of D units is determined by the scarcest of these resources. Two cases are studied, based on the distribution of the parameters of interest (i.e., user needs and price evolution): Case 1: parameters are uniformly distributed and Case 2: parameters are drawn from non-stationary process. For the evaluation, average regret over time and constraint violation over time are used as metrics.

4.4.5.1 *Main performance evaluation results*

Figure 48, delineates the convergence of algorithm OLR for the two aforementioned cases, for K = 5 slots. We observe that the average regret converges to zero and the constraint violation remains consistently below zero. In Figure 49 there is a comparison between OLR and OLR for Mixed-Time Scale (OLR-MTS) algorithms for Case 1 and 2, for different number of slots K. In OLR-MTS, we allow the policy to change the "slots" decisions within the same period, whereas in OLR, the slots decisions are committed at the beginning of the period. OLR-MTS performs better in both cases, which is rather expected, since the SP updates its decisions as new information becomes available.

Finally, Figure 50 and Figure 51 concern the OLR Slice Orchestration (OLR-SO) algorithm, where the provider is responsible for the slice composition. More precisely, in Figure 50, the convergence of OLR-SO is presented for K = 5, while in Figure 51, different values of K are taken into account. For the latter, constraint violation approaches 0 for Case 1 and has very small values for Case 2, while K increases.

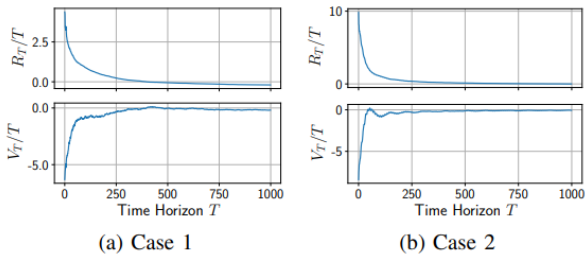


Figure 48. *Regret and violation convergence of OLR with B=10 base stations and K=5 slots.*

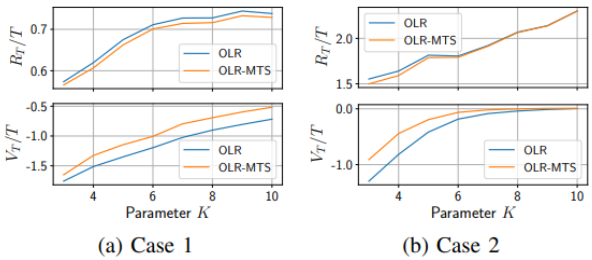


Figure 49. *Regret and violations comparison of OLR and OLR-MTS, for different values of slots K.*

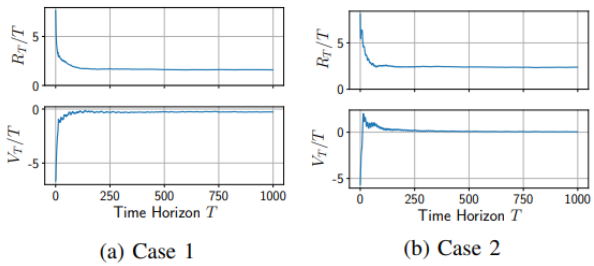


Figure 50. *Regret and violation convergence of OLR-SO with B=10 base stations and K=5 slots.*

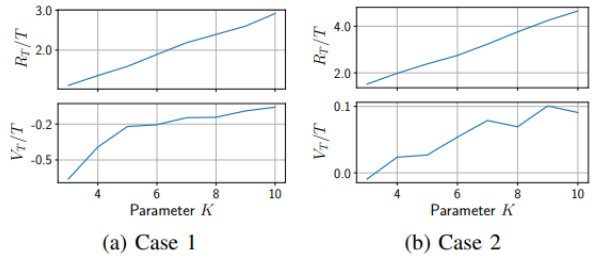


Figure 51. *Regret and violations comparison of OLR-SO, for different values of slots K.*

4.4.5.2 *Conclusions*

In this section we evaluated the performance of the proposed algorithms, for slice reservation, OLR and its variants OLR-MTS and OLR-SO. The results shows the convergence of OLR and OLR-SO in both cases analyzed. Furthermore, OLR-MTS performs better than OLR.

4.4.6 Testing EnergyEdgeCloudSim (A15)

In this section, we evaluate to what extent we can reduce energy consumption with the algorithms that we have developed for EnergyEdgeCloudSim, its impact on the number of failed requests and the scalability of our approach (K1 and K2). We construct an IoT scenario using the EnergyEdgeCloudSim simulator to evaluate our approach. To simulate the edge workload, we use the Shanghai Telcom dataset [53]. It contains six months of mobile phone records accessing the Internet via base stations distributed over Shanghai city. The data set contains more than 7.2 million records from 9481 mobile devices and 3233 base stations.

We consider an infrastructure formed by 14 edge devices with randomized characteristics. Concretely, their maximal energy consumption is between 20 and 300 Watts; the idle energy consumption (α) between 20 and 50%, P^{Tx} and P^{Rx} between 1-3 Watts; the sleep energy consumption (β) between 0.01 and 0.5; the deployment energy consumption between 0.5-1 Joules; the instructions per second of the CPU between 100000 and 300000 million; ew is 1 in all cases; the disk's capacity between 200-1000 Gb; and their RAM's capacity between 8-32 Gb (we can deactivate all nodes). This randomization considers that the most computationally powerful nodes are more energy-intensive. We make this assumption because the CPU frequency is directly related to the energy consumption. In the same way, we randomized the applications' requirements (CPU, RAM, disk, data to send and receive) and repeated experiments 30 times.

4.4.6.1 *Dynamic energy consumption*

Sometimes, other applications and users share the nodes that form the edge infrastructure, so they must always be active. Thus, this section focuses on the reduction obtained only through the new orchestrator policy, presented in Section 3.2.4. We compare the energy consumption using the Green fit policy with the energy obtained by Best fit, first fit, random fit and fastest fit policies. Best fit and First fit are included by default in EdgeCloudSim, while we have implemented Random fit and Fastest fit. In this scenario, the nodes remain active all the time, being the reduction in consumption obtained part of the dynamic energy consumption of the nodes.

Figure 52 shows the results in terms of energy consumption for the three periods of time considered and compares it with our orchestration policy (Green fit). In all cases, our policy obtains a minor energy consumption. Specifically, in our experiments, we have got a 1.6% reduction in the worst case (Random fit, 2nd hour) and up to 15.9% in the best case (14th hour) compared with Fastest-fit. As expected, the reduction in the energy consumption is significant for the 12th and 14th hours (the ones with more requests) since we focus on the dynamic energy consumption, and it is directly related to the workload. Regarding the execution time, predictably Fastest fit obtains the lowest service time on average, having Green fit and Random fit similar service times on average. Note that all the assignments accomplish the

applications' requirements in terms of QoS. The number of failed requests is 0 in all cases, as the infrastructure has resources enough to allocate the user requests.

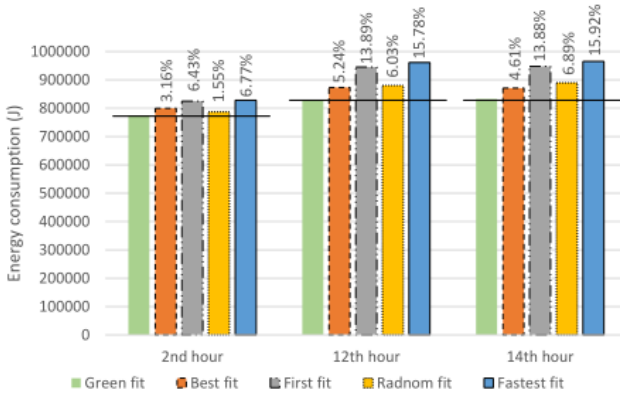


Figure 52. *Energy consumption and percentage of failed requests for each orchestration policy.*

4.4.6.2 *Dynamic and idle energy consumption*

We apply our auto-scaling approach for the same infrastructure, set of applications, and periods. We have selected the auto-scaling interval in one minute, and the orchestration policy used is Green fit.

Some workload datasets and predictive models provide information about the number of expected requests, not about the applications demanded–as is our case [54]. Given that the number of resources needed to meet these requests will depend on the required applications, we have elaborated four different resource reservation policies. Suppose we expect ten requests in the following period, and we handle six applications with an equal probability of being demanded. By likelihood, each application would be required 1.66 times, which is impossible. Considering that each application is demanded once, there will be four uncertain requests. So, we apply random assignment (it assigns the four remaining requests randomly), an oversizing (it considers that each application is demanded twice) and a most/lest resource demanding assignment. Note that our algorithm gives a solution for the worst-case scenario, in which the entire workload arrives at once right after performing the auto-scaling process.

Figure 53 shows the average energy consumption and percentage of failed requests obtained with the different resource reservation policies and operation modes. Concerning OM1 (left side), the second hour is the period with the most failed requests, with 1.5% in the best case (Most and Least resource-demanding policies, respectively). Experiments show a 24.8% decrease in the failed requests between using the Most resource-demanding policy compared with Oversizing. The least resource-demanding policy has the highest number of failed requests–and better energy consumption. Nevertheless, in the 12th and 14th hours, the Least resource-demanding policy obtains an affordable 1,5% (12th hour) and 1.8% (14th hour) of failed requests, with a decrease in the energy consumption of 14% and 13% respectively when compared with Random policy, which is the second-best in terms of energy consumption. These results mean that this policy could be a good choice in some scenarios. Regarding OM2 (right side of Figure 53), the percentage of failed requests is 0 or almost 0 in most cases, 3% in the worst case (2nd hour and Least resource-demanding policy). The energy consumption increases 26% on average in comparison with OM1

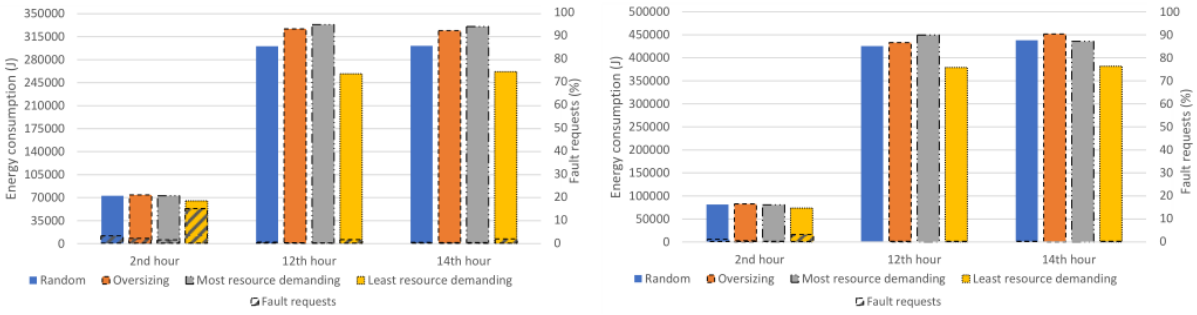


Figure 53. *Energy consumption and percentage of failed requests applying our auto-scaling approach for OM1 (left) and OM2 (right).*

4.4.6.3 *Conclusions*

Regarding energy consumption in edge-based infrastructure (K1 and K2), when the infrastructure nodes are shared with other applications and users and cannot be deactivated, we have achieved a 15.9% reduction in energy consumption. In this scenario, the more requests received, the more energy saved (compared with other policies). Applying our auto-scaling approach, we have reduced the energy

consumption by about 44% in the worst case, and up to 91% (Fastest fit compared with OM1 and Most resource-demanding policy (failed requests: 1,4%), 2nd hour) when applied to a randomized scenario. The factors that influence the lower energy consumption are the time the nodes remain idle, the lower the workload and the extended downtime.

Regarding which policy performs best, decreasing energy consumption and minimizing the number of failed deployments, the most significant reduction in energy consumption has been obtained using OM1 and the Least resource-demanding policy. Although this policy has received a high percentage of failed requests for one of the periods evaluated (2nd hour), it has an affordable 1,5% and 1.8% of failed requests in the two other periods considered. When the service must have high availability, the ENI's resource-preservative operation mode (OM2) obtains 0 or almost 0 failed requests in most cases.

4.4.7 Towards autonomous VNF scaling (A16)

The auto-scaling problem was introduced in deliverable D4.1 [46]. To recapitulate, a network application can be deployed over multiple VNFs. The workload of that application can be generated by users or from other applications. This workload enters a load balancer that distributes it according to some weights among the active VNFs. Each VNF has a First In, First Out (FIFO) queue for processing the assigned workload. When the queue is empty, the workload is processed immediately. If the workload cannot be processed, it waits in the FIFO queue until it can be processed. Additionally, a monitor constantly delivers usage metrics to a decision-making agent, which determines the amount of VNF replicas in an automatic way. The auto-scaling problem can be defined as dynamically adding or removing VNF instances to serve a variable workload [55]. In this section, we show the comparison of three auto-scaling mechanisms that do not require any information about the workload and yet can dynamically adapt the number of VNF instances while keeping them at a reasonable level without over- nor under-dimensioning the problem. The three proposed methods are a Deep Reinforcement Learning (DRL) agent based on Q-Learning, a Proportional–Integral–Derivative (PID) agent and a Threshold (THD)-based as a reactive scaler. The agent's definition details are shown in [56]. The evaluations are obtained using the simulator described in Section 3.2.1.

4.4.7.1 System scenario

A decision-making agent interacts with the simulator (i.e., environment) in regular time ticks (time steps). In practice, the agent communicates its scaling decision every tick and then waits until the monitor module generates a new report. Once a report is ready, the agent will receive it and evaluate the impact of its decisions. Moreover, the traffic generator (workload module in Figure 10) follows a known pattern in data centers, as shown in Figure 54. Generally, the traffic to a data center is low at night and peaks during working hours. This pattern repeats more or less during the weekday. The traffic is generated using:

$$W(t) = \max\Big(0, 300 \cdot (0.9 + 0.1\cos(\pi \cdot T/10)) \cdot \big(4 + 1.2\sin(2\pi \cdot T) - 0.6\sin(6\pi \cdot T) + 0.02(\sin(503\pi \cdot T) - \sin(709\pi \cdot T))\big)\Big) + 5N(t) + I(t)$$

Where $T = \frac{t}{86400}$, which re-expresses the time t expressed in ticks (i.e., seconds) in T days, the term $\sin(2\pi \cdot T)$ introduces a daily pattern and $\sin(6\pi \cdot T)$ an 8h pattern. The rest of the terms introduce some randomness so that this pattern does not repeat itself every day. In particular, $N(t)$ is a zero-mean, unit-variance Gaussian random variable and $I(t)$ introduces exponentially decaying impulses on average every 10 000s of average height 200 jobs lasting about 500s.

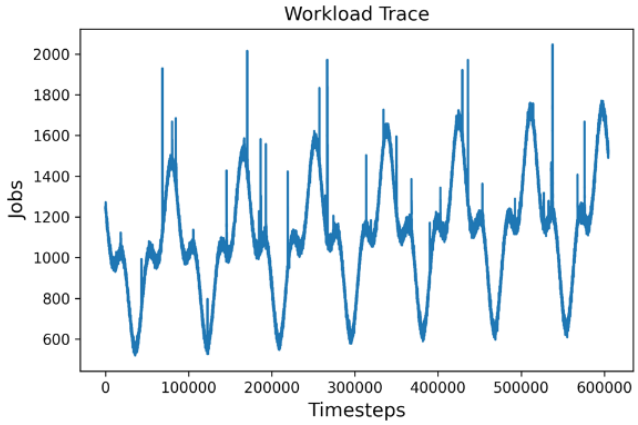


Figure 54. *Complete workload trace used in activity A16.*

4.4.7.2 Models implementation

We implemented our DQN agent using Stable-Baselines3 (SB3) [57], a framework that implements popular RL algorithms in Pytorch[12]. In the definition of the DQN agent, we used the default values given by the SB3 framework. During training, the agent tries to maximize a reward function, if the peak latency or the CPU usage of the active VNFs are within a tolerance range, the agent is rewarded; otherwise, the agent is not rewarded. Typically, the agent is more likely to take actions that produced a reward in the past by taking the actions that led to that situation (exploitation). However, the agent must take random and possibly new actions (exploration) to discover the actions that maximize its reward.

On the other hand, the PID agent tries to keep the peak latency around $d_{tgt} = 20ms$. The optimal values for its parameters α and β were determined by an exhaustive search. The parameter space (α, β) was sampled by letting α range over the values {0.125; 0.25; 0.5; 1; 2; 4; 8} and β over {50; 100; 200; 400}. Then it was determined for which of all these combinations the latency was the least amount of time above the tolerated upper bound of $(1 + \epsilon)d_{tgt}$, when the PID agent controls the first part of the workload trace, i.e., the training set. It turns out that if the training set spans the first day, the optimal parameters are $(\alpha, \beta) = (16, 200)$, while if the training set spans the first two days, the optimal parameters are $(\alpha, \beta) = (0.25, 200)$. In both these cases, the minimum is broad: relatively small changes in α and β do not alter the number of latency violations drastically so that the choice of α and β is not critical.

4.4.7.3 Main results

To test the agents' behavior in unseen workload traces, they were tested using the last 172.8K workload values. It is important to notice that the DQN (and THD-based) agent and the PID agent use different information as input. The former uses the instant peak latency and CPU load, while the latter uses the instant and previous peak latency. Also, the RL agent learns automatically, while the PID agent is manually tuned. Both of these facts mean that care should be taken when comparing the performance of these agents.

Table 20. *Comparison results of DQN, THD and PID agents in terms of number of VNFs and peak latency.*

Metric	Approach	Mean	Std	Min	25%	50%	75%	Max
Number of VNFs	DQN	4.87	0.84	1	5	5	5	8
	THD	4.32	1.25	1	3	4	5	17
	PID	4.06	1.09	1	3	4	5	10
Peak Latency [s]	DQN	0.0095	0.0025	0.0058	0.0088	0.0090	0.0093	0.0785
	THD	0.0153	0.007	0.0058	0.0099	0.0118	0.0195	0.1432
	PID	0.0198	0.0048	0.0033	0.0163	0.0194	0.0228	0.0689

Table 20 gives a quantitative analysis of the behavior by showing the main statistical figures: mean, standard deviation, minimum, maximum, and the most representative quartiles of the peak latency and number of created VNFs. As can be seen, the DQN can maintain a more stable number of created VNFs than the PID and the THD-based agents. However, this is more a secondary effect since all the agents are not designed to optimize the number of replicas. Regarding the peak latency, most of the time, all the agents can keep this metric under the upper bound (24ms). Nonetheless, as shown in **Table 21**, the PID agent violates the upper bound 16.57% of the time while, the THD-based and the DQN are reducing the violations to 12.2% and 0.69%, respectively.

Table 21. *SLA Violations from the DQN, THD and PID agents.*

Approach	% SLA Violations
DQN	0.69%
THD	12.20%
PID	16.57%

4.4.7.4 Conclusions

We designed and evaluated three autonomous scaling agents using known techniques such as heuristics, classic control and RL. We compared the three agents in terms of the peak latency, the amount of created VNFs, and the amount of SLA violations. However, choosing the applicable agent is a task beyond only performance evaluation. It also depends on both business- and operational-related conditions. On the one hand, a multi-tier SLA between stakeholders might show different amounts of marginal penalty among agreed objectives, e.g., a high penalty even when slightly violating the maximum service latency; therefore, the auto-scaler agent may have a higher chance to disregard the

[12] https://pytorch.org/

number of created VNFs. On the other hand, from the operational point of view, an operator might not have the required hardware to support ML solutions. Therefore, a DRL agent is ruled out due to its requirement to explore new actions to improve the reward, leading to unpredictable behavior on lower-end hardware.

4.4.8 Auto scaling Virtualized RAN caches (A17)

In this activity, we evaluation the elastic femtocaching algorithms proposed in Section 6.2.2 of D4.1 [46], by assessing their performance in a series of experiments with real data traces (D9).

4.4.8.1 Elastic femtocaching evaluation

Firstly, for the topology of BSs, we considered two cases in the simulations: (1) linear BS topology with manual parameters; (2) real BS topology of a mobile operator on the west side of the US for rural, suburban and urban areas, as shown in Figure 55, with real parameters. In order to analyze the performance of the proposed General Algorithm with Joint Cache Rental and File Caching (GA+JCC) and with Joint Cache Rental, File Caching and Routing (GA+JGCA), sum delay utilities of all subareas are employed. The aforementioned elastic algorithms are compared with four known static algorithms, namely Fixed Cache Lease Budget (FCB), Fixed File Caching (FFC), Least Recently Used (LRU), multi-LRU and q-LRU and Least Frequently Used (LFU).

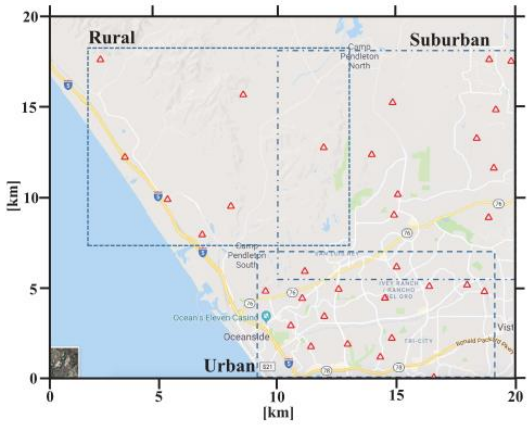
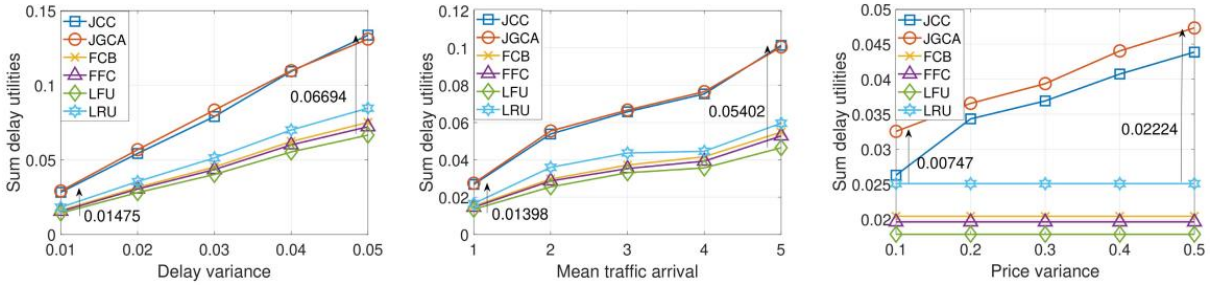


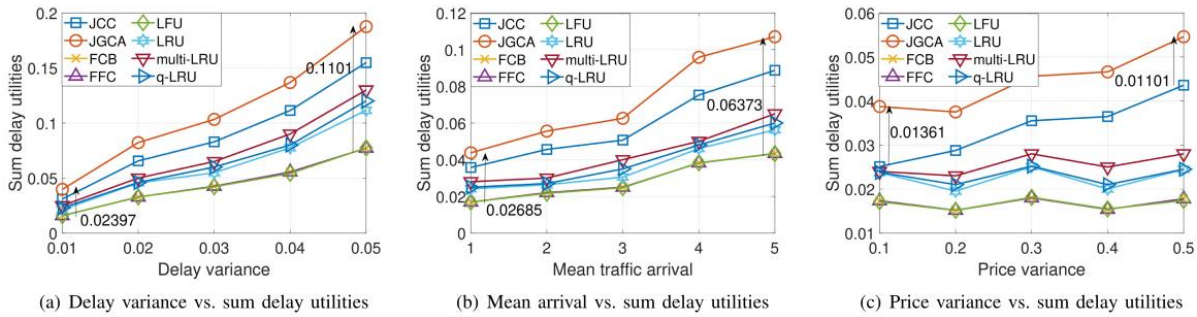
Figure 55. *Rural and Suburban Deployment of BSs used in evaluation of vRAN rescaling Algorithm.*

To delineate the impact of different parameters (variance of delay, mean traffic arrival, and variance of pricing) on the system performance, we first show the simulation results in the linear BS topology, under two scenarios: non-overlapping SBSs and overlapping SBSs. In the former case, as can be viewed from Figure 56, GA with JCC is an optimal algorithm since the subarea-SBS association and content caching is uncoupled. At the same time, GA+JGCA algorithm achieves a similar performance with the optimal and they both outperform the existing static cache leasing algorithms in the case that the network environments and pricing drastically change (i.e, the variation of input parameters becomes higher and mean traffic arrival increases). For the latter case, GA+JGCA is the optimal algorithm since the subarea-SBS association and content caching are tightly coupled with each other. Similar to the previous scenario, the elastic cache leasing policies (GA+JCC and GA+JGCA) are much better than the existing static cache leasing policies (FCB, FFC, LFU, LRU, multi-LRU, q-LRU). Results are depicted in Figure 57.



(a) Delay variance vs. sum delay utilities (b) Mean arrival vs. sum delay utilities (c) Price variance vs. sum delay utilities

Figure 56. *Sum utilities in linear BS topology case and non-overlapping SBSs scenario.*



(a) Delay variance vs. sum delay utilities (b) Mean arrival vs. sum delay utilities (c) Price variance vs. sum delay utilities

Figure 57. *Sum utilities in linear BS topology case and overlapping SBSs scenario.*

Lastly, in Figure 58 the performance gain (i.e., the gain of sum delay utilities) of the proposed GA+JGCA algorithm over static FCB algorithm and the proposed GA+JCC algorithm which uncouples routing and caching decision is shown, for the real BS topology. There, the delay profile of each user from each BS is more heterogeneous. Hence, for denser BS topologies (urban area), the impact of the elastic cache leasing policy (i.e., GA+JGCA) on the system performance increases. Finally, joint control of cache leasing, file caching and routing becomes more important as BS topology becomes denser. This interpretation can be driven from the fact that as BS topology gets denser (i.e., from rural area to urban area), the gain from the routing-caching the uncoupled solution, i.e., GA+JCC to the joint solution, i.e., GA+JGCA becomes higher.

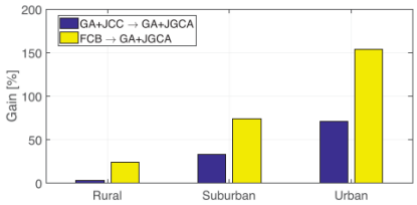


Figure 58. *Performance gain of the proposed algorithms under real BS topologies and SBSs scenarios.*

4.4.8.2 *Conclusions*

In this section we evaluated the performance of two elastic femotchaching algorithms, namely GA+JCC and GA+JGCA, by means of a comparison with four different static algorithms. Our results shows that our proposed algorithms perform better than the static ones, which are outperformed in case the network environment and pricing drastically change. Furthermore, GA+JGCA shows a performance gain increase, against static algorithms and GA+JCC, as the BS deployment increases.

4.5 NI for automated anomaly response

Evaluation E5 aims at evaluating NI solutions for anomaly response. The DAEMON consortium performed assessments of challenges and solutions related to E5 via activity A18 and A19. **Table 22** summarizes the tools, KPIs, TRL, PoC plans, approximate progress and main innovations of such activities.

Table 22. *List of activities for E5.*

ID	Name	Evaluation	Tool	Planned KPIs	Collected KPIs	Target TRL	Planned for PoC demo	Progress
A18	In-backhaul learning	**E5**	D7	K3	None[13]	2	TBD	5%
	Main innovation: Inference fully performed in the user plane, at line rate, via programmable switches							
A19	Anomaly detection for a roaming platform	**E5**	D8	K3, K7	K7	5	Yes	60%
	Main innovation: Device-level anomaly detection for IoT verticals							

[13] At this level of the development, we have not collected any KPI yet for A18.

Overall, the preliminary results of these activities already led to key observations on potential approaches to NI-assisted anomaly detection in future-generation mobile networks, as follows.

- ***We demonstrate that traditional Random Forest models perform as a well as more complex neural networks in classification and anomaly detection tasks in highly constrained user-plane environments***. The results of ***A18*** pave the road for future steps in the project towards developing NI models that are suitable for integration in programmable user planes. We will build on these insights to design and implement NI models for line-rate inference during the second iteration of the project.
- ***We show that NI models based on deep learning approaches can identify anomalies in the network signaling data for roaming operations, which are not reported by legacy systems***. To achieve this result, we developed a methodology of measurement data analysis and clustering, which allowed testing NI models with unprecedented real-world ground-truth data. The activities in A19 thus pave the road for NI-assisted generation of alarms that the operations team of the mobile network can study in real-time. The deployment of such solutions in a pre-production system will be at the core of the work during the second iteration of the project for this activity.

As anticipated in Section 4.3, the consortium will also consider the possibility of merging E3 and E5 during the second iteration of the project. This will be reflected in the next deliverable of WP5 in case the option concretizes.

4.5.1 In-backhaul learning (A18)

DAEMON's architecture addresses the challenge of meeting the very stringent requirements of beyond 5G services, in terms of throughput and latency, at different levels of the edge to core continuum. In particular, we bring intelligence at the Transport level, to achieve a 1-ms response time of the NI algorithms, targeting KPI K3. In this section, we present our study and our initial implementation of the inference phase of supervised classification directly into the user plane of programmable switches. Our study and considerations on the challenges represented by such implementations, due to hardware limitations of the programmable switches, are described in section 7.1 of Deliverable 3.1 [34].

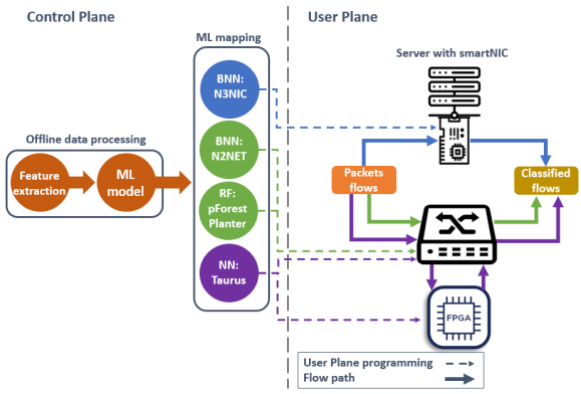


Figure 59. *Summary of the different approaches for in-backhaul inference.*

4.5.1.1 In-backhaul inference approaches

Given the limitations of programmable switches and smartNICs, all existing approaches in scope of our study assume that computationally expensive training is performed offline; the problem is then deploying the trained model entirely in the user plane, so as to achieve line-rate operation. **Figure 59** offers a comprehensive view of the workflow adopted by different proposed approaches to address such a problem. Common to all approaches is a second stage in the control plane, where the trained model is encoded for operation in the user plane, e.g., by rendering it via a network programming language such as P4. Where proposals vary is in the family of machine learning models considered, and in the nature of the programmable hardware targeted. In the following, we describe 4 different approaches.

i. Decision tree-based models on switches: Decision tree (DT) or Random Forest (RF) models [58, 59, 62, 63], with a relatively low complexity, are deployed the complete ML in a single off-the-shelf switch. Hence, as shown in **Figure 59**, solutions like Planter [59] or pForest [58] let P4-programmed DT and RF models process packets at line-rate using solely the match-action pipelines of the switch. The different approaches are told apart by the way they map tree structures to match-action tables.

ii. Neural networks on Switches: more complex DNN models have also been considered for in-switch implementation, with the sole example of N2Net [64]. N2Net uses Binarized Neural Network (BNN) models, which rely on +1/−1 weights and sign activations, enjoying a much reduced memory footprint, thus more suitable for in-switch implementation than regular DNNs. As

illustrated in **Figure 59**, the workflow is the same of DT and RF models above. The difference is the mapping of the model to the match-action tables, which is specific to BNN architectures. In fact, it is important to stress that even BNNs are onerous to deploy in-switch: a basic BNN with two layers of 64 and 32 neurons would exhaust completely the resources of a Tofino ASIC [64].

iii. Neural Networks on SmartNICs: BNNs are integrated on SmartNICs. N3IC [65] realizes the integration using both the micro-C language (for Netronome system-on-chip NICs) and P4 (for NetFPGAs NICs configured with a PISA architecture). **Figure 59** highlights how N3IC maps the ML model to a SmartNIC hardware located in a server. The SmartNIC environment offers a good amount of computational resources and increased memory size, which allow to implement a 3-layer BNN that operates at line-rate. Yet, SmartNICs are deployed at network appliances that reside in a host within the network datacenter, thus granting inference at specific locations of the network only, and not at any point of the transport domain as with in-switch solutions.

iv. Neural networks on custom switches: dedicated hardware is added to switches or smartNICs to implement complex DNN models. Taurus [66] framework employs a custom accelerator to implement DNNs via flexible MapReduce operations. Taurus-enhanced switches grant complete freedom in the deployment of ML models in user planes. However, Taurus' adoption at scale would require revisiting the user plane design of network transport domains, deploying significant custom hardware next to already expensive programmable switches and smartNICs.

4.5.1.2 *Comparative evaluation of ML models*

We compared the different studies presented and assessed the performance of the solutions they propose with assorted use cases and diverse traffic datasets. We tested the performance of the DT, RF, DNN and BNN models parametrized according to the indications in the original works.

To ensure maximum fairness of the evaluation, we produce results for all use cases (that are publicly available) explored in the original works presenting each inference model. This results into two traffic classification tasks and three anomaly detection tasks, which correspond to the datasets D7, D11, D12, D13, D14 described in Section 3.3. The results are summarized in

Figure 60. Results of the comparative evaluation of machine learning models used for in-band inference. The best result for each use case and metric is highlighted in bold, the second best in blue. and are quite manifest, as the DT, RF and NN models achieve performance that can be considered satisfactory across all use cases, with F1-scores typically in the 95 − 100 range, and consistently good in all other metrics as well. BNNs lag instead behind with less consistent and lower accuracy. A second takeaway is that all metrics show nearly identical values under full and early flows approaches: since computing features on early flows grants anticipated classification of traffic or detection of anomalies, the second strategy is largely preferable in all considered use cases. Thirdly, and most importantly, RF emerges as the overall winner of our comparative evaluation. The model is often the best performing one, or is a close second otherwise.

			Accuracy		Precision		Recall		F1-score	
Use case	**Dataset**	**Model**	**Full flows**	**Early flows**	**Full flows**	**Early flows**	**Full flows**	**Early flows**	**Full flows**	**Early flows**
Traffic Classification	UNSW-IoT	DT(10)	96.254	97.771	96.617	97.717	95.292	97.628	95.896	97.645
		RF(10,5)	**98.758**	**98.695**	**98.776**	**98.984**	**98.499**	**98.593**	**98.635**	**98.779**
		NN(128,64,6)	97.249	97.362	97.336	97.361	97.249	97.362	97.249	97.338
		BNN(128,64,6)	72.267	83.002	74.481	84.435	72.267	83.002	70.858	82.909
	UNIBS-2009	DT(9)	99.359	99.737	93.499	**96.345**	**93.931**	**97.406**	93.706	**96.851**
		RF(9,5)	**99.532**	99.521	**96.091**	95.077	92.955	95.107	**94.280**	95.078
		NN(64,32,8)	97.838	99.223	90.965	95.261	92.471	89.116	91.627	91.129
		BNN(64,32,8)	85.501	89.747	80.063	90.479	85.501	89.746	80.792	87.169
Anomaly Detection	CICIDS-2017	DT(10)	**99.915**	99.744	**99.924**	99.799	99.824	99.590	**99.874**	99.694
		RF(10,5)	**99.915**	**99.762**	**99.924**	**99.805**	99.824	**99.628**	**99.874**	**99.716**
		NN(128,64,2)	99.865	99.259	99.691	99.263	**99.909**	99.259	99.799	99.260
		BNN(128,64,2)	97.748	89.807	95.224	87.305	98.563	92.622	96.765	88.802
	UNSW-NB15	DT(7)	98.712	98.883	**86.768**	90.177	94.957	92.467	90.417	91.288
		RF(10,5)	**98.772**	**99.213**	86.398	91.492	**97.655**	97.055	**91.206**	**94.084**
		NN(128,64,2)	98.533	98.971	85.628	88.056	93.255	**99.029**	89.040	92.789
		BNN(128,64,2)	94.284	91.587	64.236	62.405	78.293	87.599	68.528	67.214
	NSL-KDD	DT(10)	**97.037**	–	**97.015**	–	**97.071**	–	**97.035**	–
		RF(10,5)	96.990	–	96.973	–	97.004	–	96.986	–
		NN(12,6,3,2)	91.984	–	92.275	–	91.830	–	91.939	–
		BNN(12,6,3,2)	85.254	–	87.406	–	84.770	–	84.895	–

Figure 60. *Results of the comparative evaluation of machine learning models used for in-band inference. The best result for each use case and metric is highlighted in bold, the second best in blue.*

4.5.1.3 *Conclusion and outlook*

Our comparative analysis shows that RF models based on early flows are a promising candidate for deployment of machine learning in the user plane, assuming that they can be efficiently integrated in off-the-shelf programmable hardware. Whether this is the case is an open question that we aim to dispel in the second part of our verification study. We plan to implement RF models in P4 and evaluate their performance in bmv2 software switches and on real hardware, by exploiting testbed T12, described in Section 3.1.12, and to report such results in the next deliverables of WP5.

4.5.2 Anomaly detection for a roaming platform (A19)

In this activity, using the dataset D8 we previously introduced, we aim to test the performance of the anomaly detection models we introduced in deliverable D4.1 [46]. Given the heterogeneity of signaling behavior corresponding to the IoT devices we monitor, our methodology includes a clustering step that allows us to run the anomaly detection models on devices with similar behavior under non-anomalous circumstances. We detail this step next.

4.5.2.1 Clustering devices

We run the clustering algorithm on the test set to identify to which cluster each device belongs, and train and apply the anomaly detection algorithm on each cluster independently. For the December 1st, 2019 – January 12th, 2020 period, the clustering algorithm divides the devices of the customer in three groups accounting for 79%, 17% and 4% of the total amount of active devices, respectively. We report on the distribution of the signaling volume of each cluster in Figure 61, where we show the distribution of the average amount of daily messages per device. We observe that the third cluster is the one with highest signaling frequency (with about 2,000 daily messages generated per device on average), and that devices belonging to cluster one and two generate 24 and 215 average daily messages, respectively.

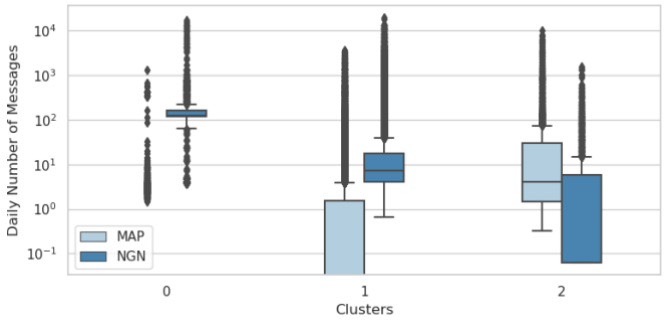


Figure 61. *Clusters of devices: We find three groups of devices containing different amount of signaling traffic each. Groups are well defined as the upper and lower quartiles of the boxplots do not overlap between them in the vertical axis.*

For each cluster, we obtain a ranked list of anomalies from each model, where top-most devices correspond to detected anomalies. In the GMM model, devices are ranked based on the probability of pertaining to the Gaussian distribution (lower values first). A low probability means the device behaves different from the rest, likely being an anomaly. Regarding VAE models we use the KL divergence, which corresponds to the difference between the learned latent space and the normal distribution $N(0,1)$. 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 of being an anomaly.

4.5.2.2 Models performance evaluation

To evaluate our models, we built a meaningful and representative ground truth dataset by collecting trouble SIMs from the network ticketing system (incidences occurred during our testing period) and compute the accuracy of detecting those SIMs by each of four models. Our goal is not only detecting already known incidences but missed anomalies that were not registered in the alarm system, but because somebody reported them.

Due to the manner in which we choose to run our clustering, and also represent our data, we are able to capture SIMs with aggressive behavior in terms of signaling. Specifically, in Table 23 we show a few examples of tickets that reported anomalies related to the behavior of several SIMs active in Belgium. We note that our deep learning models were able to capture these anomalies by generating a high value of the KLD metric.

Table 23. *Examples of anomalies we collected from the ticketinig system of the IPX operations team.*

Devices	Tickets date	Description
10 SIMs in Belgium	2020-02-10	Aggressive sim behaviour against **Belgium** radio network.
5 SIMs	2020-02-10	URGENT (Recurrent) Ticket: Data connection is blocked for the device, cannot establish data communications because of a erroneous alarm of high data consumption.
2 SIMs	2020-02-10	Aggressive behavior of devices against the radio network provider in Belgium. One device stopped the aggressive signaling, while the other continues.
1 SIM	2020-02-11 2020-02-23	Long PING delay reported as an anomaly by the Client, was actually the normal behavior for M2M SIM card and roaming scenario and works as expected.

10 SIMs in Belgium	2020-02-23	Aggressive signaling behaviour of devices in Belgium.
1 SIM in Canada	2020-02-23	Client trying to test one device in Canada using the indicated SIM, but the device is not coming online.

4.5.2.3 *Conclusions and future directions*

Overall, we notice that the DL models are able to capture the aggressive signaling behavior of the IoT devices in our dataset. We especially mention that our tool was able to draw the attention of the operations team to the aggressive behavior of device in Belgium. This type of anomalies are indeed very harmful to the local operator in Belgium, and as well to the IPX operator. In fact, the devices that our approach tagged as anomalous were to blame for compromising the partnership agreements between the home network of the SIMs and the local operator in Belgium.

Our next effort goes towards validating with the platform operations team a set of anomalies that only our anomaly detection approaches were able to capture. For this, we monitor the evolution of the KLD anomaly score to generate alarms for the operations team to study.

4.6 NI for capacity forecasting and self-learning

Evaluation E6 focuses on the evaluation of NI solutions for long-timescale operations, i.e., MANO, VNF placement and the associated resource allocation. The DAEMON consortium performed assessments of challenges and solutions related to E6 via activity A20 and A22. **Table 24** summarizes the tools, KPIs, TRL, PoC plans, approximate progress and main innovations of such activities.

Table 24. *List of activities for E6.*

ID	Name	Evaluation	Tool	Planned KPIs	Collected KPIs	Target TRL	Planned for PoC demo	Progress
A20	Anticipatory capacity allocation	**E6**	D1	K2, K4	K2, K4	3	No	60%
	Main innovation: Anticipatory provisioning of network resources to individual network slices so as to avoid underprovisioning while minimizing the unnecessary allocation of resources							
A21	Virtual Machine reservation	**E6**	D1	K2, K9	K2, K9	3	No	40%
	Main innovation: Prediction of 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							
A22	Minimization of video streaming slice OPEX	**E6**	D1	K4, K9	K4, K9	3	No	40%
	Main innovation: Minimization of the monetary OPerating EXpenses (OPEX) associated to running the video streaming slices at the network Edge							

Overall, the preliminary results of these activities already led to observations on the performance of NI solutions that can anticipate the allocation of resources in future-generation mobile networks, as follows.

- ***We prove how hybrid NI design that combine statistical modelling and machine learning can outperform pure deep learning approaches in traditional resource allocation tasks***. The activities carried out in ***A20*** showcase this in practical settings and with large-scale measurement data, hence supporting further investigations of such a NI design strategy during the second iteration.
- ***We demonstrate that automating the design of loss functions for deep learning models can largely benefit anticipatory networking tasks***. Extensive tests in realistic settings and against state-of-the-art benchmarks demonstrate the viability of this concept, as well as the potential performance gains it can unlock. Both activities ***A21*** and ***A22*** contribute to the evaluation of such an original model, setting forth important contributions towards practical IBN systems. The results of A21 and A22 thus pave the road to refinements and enhancements of the loss-learning paradigm during the second iteration of the project, whose results are expected to be presented in the next deliverable of WP5.

4.6.1 Anticipatory capacity allocation (A20)

This activity targets a capacity forecasting scenario where anticipatory NI is in charge of the allocation of network (e.g., compute, transport, memory) resources to individual network slices. Here, it is critical

that the capacity prediction avoids all underestimation (which causes allocation of insufficient capacity to slices, hence disruption of the service experienced by the end user) while minimizing overprovisioning (determining an unnecessary allocation of resources that will ultimately go wasted).

The NI solution developed by the project to address this problem is named TES-RNN, and is outlined in Section 4.1 of Deliverable 4.1 [46], where formal definitions of the target system and problem are also provided. Here, we detail the evaluation settings and performance results.

We set the capacity allocation use case in a network core Cloud scenario, where a single large datacenter runs VNFs for the traffic generated in the whole target region by four traffic-intensive mobile applications, i.e., Facebook, Instagram, Snapchat and YouTube. The traffic demands for each service are derived from dataset D3 described in Section 3.3.3. We assume that each service above is assigned a dedicated network slice, and that the NI responsible for capacity allocation at the datacenter must reserve in advance enough resources to accommodate the future demand of single slices. Therefore, this setup allows evaluating how forecasting models such as the one we propose can assist NI in a multi-service and multi-slice environment. In the following, we compare our proposed TES-RNN model against three relevant benchmarks, as indicated in Section 4.1 of Deliverable 4.1 [46].

4.6.1.1 Overall capacity forecasting performance

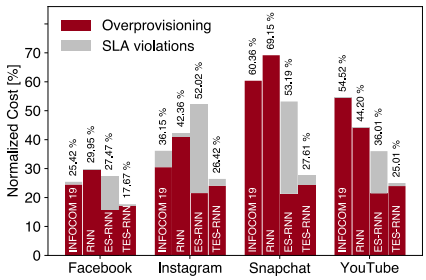


Figure 62. *Additional capacity allocation cost caused by INFOCOM19, RNN, ES-RNN, and TES-RNN prediction errors. Results refer to four slices assigned to specific services at a network core datacenter.*

We start by comparing the total costs incurred by the operator when using the different forecasting models to support capacity allocation, in Figure 62. Costs are expressed as the percent excess over a baseline given by an oracle that makes a perfect prediction, telling apart the fraction of the cost resulting from resource overprovisioning and SLA violations. The key observation is that TES-RNN consistently outperforms the benchmarks, with gains over the second-best solution that range between 8% and 25%, as well as very low SLA violation probabilities.

4.6.1.2 In-depth analysis of one prediction instance

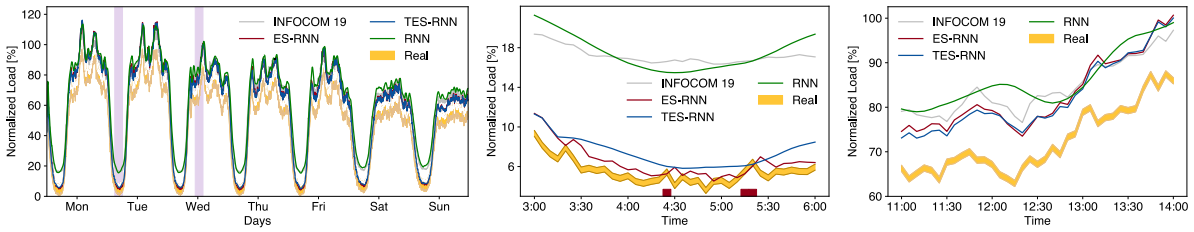


Figure 63. *Time series of the real traffic of the Facebook slice, and of the relative capacity predictions of INFOCOM19, RNN, ES-RNN and TES-RNN. Left: weekly time serie. Center: view of the 3:00-6:00 interval of Tuesday, with SLA violation periods of ES-RNN in red. Right: zview of the 11:00-14:00 interval of Wednesday.*

To gain additional understanding on the behaviors of the forecasting models presented above, we detail a representative case of capacity prediction in Figure 63. The plots show the time series of the real traffic in the Facebook slice, as well as the corresponding capacity allocation foreseen by each predictor.

The left plot portrays the traffic dynamics over a full week, and underscores how all models follow well the long-timescale fluctuations of the demands, such as low overnight traffic or different activity peaks during daylight. Center and right plots present a close-in view of two specific 3-hour periods, which are evidenced by vertical shades in the left plot. The zoom magnifies how TES-RNN and ES-RNN help dimensioning a capacity that is closer to the real demand than that anticipated by INFOCOM19 and RNN, especially in low traffic conditions.

4.6.1.3 *Control of SLA violations*

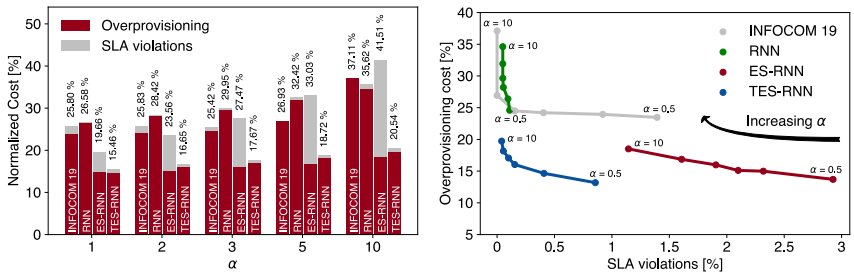


Figure 64. *Left: Additional capacity allocation cost of INFOCOM19, RNN, ES-RNN, TES-RNN prediction errors, versus α and for the Facebook slice. Right: Limits in terms of SLA violations and overprovisioning costs that can be attained by TES-RNN, and the chosen benchmarks for the Facebook slice.*

The results presented before are for one specific value of the parameter α that controls the equilibrium of overprovisioning and SLA violation risk in the considered loss function (see Section 4.1.1 of Deliverable 4.1 [46] for details). The left plot in Figure 64 illustrates the capability of each model to enforce the desired control above, for the case of the Facebook slice. We observe that TES-RNN yields again the best performance in all settings. More importantly, it keeps the overall cost low by progressively decreasing the occurrence of SLA violations as a grows, which is exactly the desired behavior. INFOCOM19 and RNN can also achieve this result, however at a cost in terms of overprovisioning that is almost twice that of our hybrid model. ES-RNN is instead unable to modulate the SLA violation cost, which in fact grows with α.

The right plot of Figure 64 gives a view of the operating points of each forecasting method. TES-RNN offers the best options to the operator, as it allows choosing among configurations that simultaneously provide less SLA violations and lower overprovisioning costs than the benchmarks.

4.6.1.4 *Conclusions*

By enhancing a recently proposed method for joint optimization of statistical models and neural network architectures, a hybrid model such as TES-RNN offers significant performance gains in anticipatory networking tasks. When confronted with a practical application use cases characterized by real-world traffic volumes and dynamics, TES-RNN granted gains up to 25% over state-of-the-art predictors that were specifically designed for the target problem. These results lay solid foundations to further research on hybrid approaches to NI design, which will be explored during the second iteration of the project.

4.6.2 Virtual Machine reservation (A21)

This activity focuses on a capacity forecasting use case centered around a core network datacenter setting. There, different video streaming services are assigned individual and dedicated Network Slice Subnet Instances (NSSI). The Virtual Infrastructure Manager (VIM) responsible for controlling 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. The problem consists in developing a suitable NI to manage VM reservations to accommodate future demands for each NSSI or, equivalently, service.

In order to perform our experiments, we employ traffic demands for each video streaming service from Dataset D3 described in Section 3.3.3. Also, we emulate the actual operating costs of the datacenter and the local VM management strategies with a realistic expression $f_{\mathcal{M}}$ for the management objective. Note that such an expression may not be (fully) known by the network operator a priori, in which case the NI needs to learn $f_{\mathcal{M}}$ from experience, i.e., by observing how the system responds to VM reservations over time. The solution we devise for the NI algorithm is the Loss Learning Predictor, or LossLeaP, whose architecture is detailed in Section 4.2 of Deliverable 4.1 [46], where more information on the expression $f_{\mathcal{M}}$ used in the tests are also provided.

We assume that the VM orchestration takes place every 5 minutes, which is thus the forecast horizon of the predictor. We feed LossLeaP with past traffic information, and let it (i) learn an approximation of $f_{\mathcal{M}}$ and, jointly, (ii) learn to produce a forecast d_t that minimizes such $f_{\mathcal{M}}$. We consider two benchmarks for comparison, as follows.

- An Oracle predictor, which has perfect knowledge of the future and always allocates the optimal minimum number of VMs to serve the upcoming demand.
- A legacy recurrent neural network (RNN) forecasting model trained to minimize a Mean Square Error (MSE) loss. A downstream decision-making block uses the forecast to make VM reservations. Details on the decision-making block are in Section 4.2 of Deliverable 4.1 [46].

4.6.2.1 *Comparative performance summary*

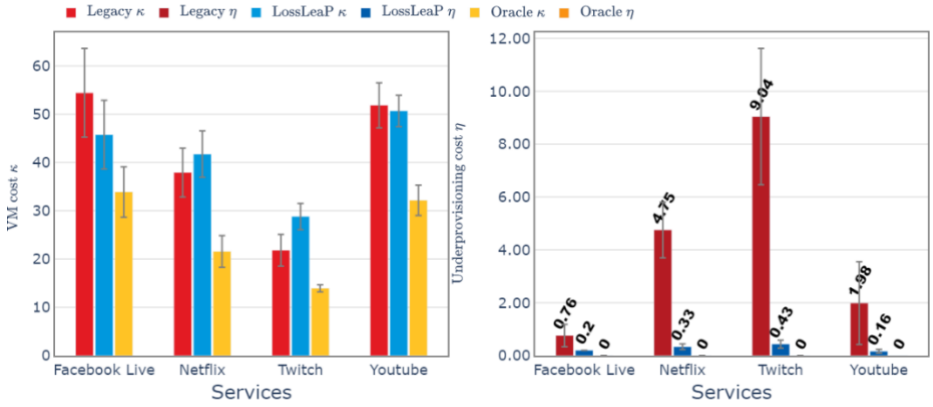


Figure 65. *VM reservation for diverse slices. Left: reserved VMs. Right: fraction of time when the slice demand cannot be served.*

Figure 65 summarizes the performance of LossLeaP in the considered case study. Even when fine-tuned, a decision-making policy based on a legacy prediction is substantially less efficient than LossLeaP. Our model allocates around the same VMs as legacy, but with an extremely limited under-provisioning that is one or two orders of magnitude lower than that of legacy.

4.6.2.2 *In-depth analysis of model performance*

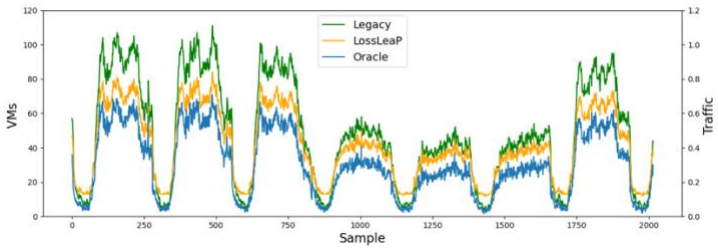


Figure 66. *Example of VM reservation for the Facebook NSSI.*

The reason is illustrated in Figure 66, for one sample NSSI, i.e., the Facebook video streaming service: LossLeaP anticipates a constant overdimensioning at all times; instead, the solution based on the Legacy predictor tends to allocate excess VMs during the high-traffic daylight hours, and does not leave a wide enough safety margin overnight, when it allocates less VMs than Oracle, hence not servicing part of the demand. Instead, LossLeaP learns that a static overprovisioning factor is not a good strategy to cope with the inherent forecast inaccuracy, and automatically identifies a better loss to minimize the (unknown) expression of $f_{\mathcal{M}}$. By doing so, our proposed NI performs in fact fairly close to the Oracle.

4.6.2.3 *Conclusions*

Automating the design of loss functions for anticipatory reservation of VMs allows achieving significant gains, with a 4x to 20x reduction of SLA violations due to the underprovisioning of VMs in a network core datacenter. Our results are obtained in realistic settings and against a state-of-the-art benchmark. Building on the excellent results achieved in this use case, we plan to refine and make more robust the design of loss-learning architectures for NI in the second iteration of the project.

4.6.3 Minimization of video streaming slice OPEX (A22)

This activity casts the capacity forecasting problem in a mobile Edge environment where computational facilities serve a large number of individual radio access base stations located in their proximity, so as to reduce latency in service provisioning. As in the case of A17 presented in Section 4.6.2, we assume that each of four video streaming services has a dedicated NSSI at the Edge facilities, and we derive realistic traffic demands for such services from Dataset D3 described in Section 3.3.3. Here, the management goal is minimizing the monetary Operating Expenses (OPEX) associated to 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.

The ground-truth OPEX is emulated via a complex numerical model that takes advantage of real-world measurements from studies in the literature and that relates the OPEX to the system variables. It is worth noting that also in this case the expression of the objective involved and typically not known to the operator in practical cases, since, e.g., the OPEX depends on the QoE and MOS of end users in entangled

ways. This makes LossLeaP a suitable forecasting model to support a NI for compute resource allocation in the target scenario. Details on the system and OPEX models are in Section 4.3 of Deliverable 4.1 [46].

We compare the proposed LossLeaP solution against two benchmarks, as follows: (i) an Oracle predictor that knows the future and OPEX model, and returns an optimal allocation; (ii) DeepCog, a state-of-the-art capacity predictor [60] for capacity forecasting, whose manually designed loss function is configured with prior knowledge of the OPEX model.

4.6.3.1 *Comparative performance summary*

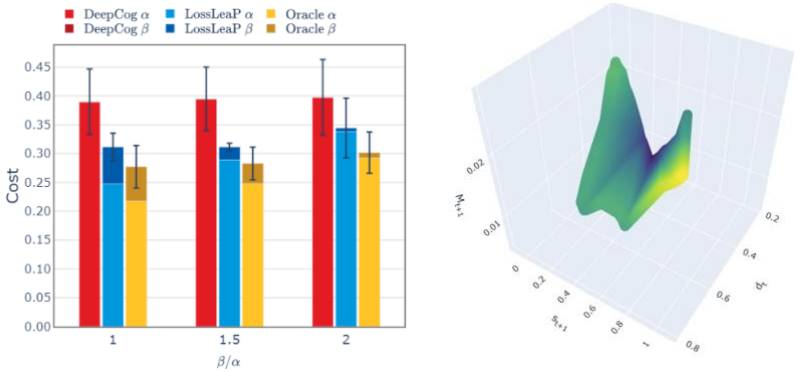


Figure 67. *OPEX performance in the Facebook Live slice case. Left: overall cost. Right: loss function learned by LossLeaP.*

For the sake of brevity, we only show results for the Facebook Live NSSI, in the left plot of Figure 67, yet performances are homogeneous across services. All costs are relative to that of the minimum static capacity allocation that always services the full demand, and are shown for different parametrizations of the OPEX model (along the x axis). Despite the fact that we feed it with information about the true parameters used in the OPEX model, DeepCog is still constrained by its unflexible and manually designed loss function. Instead, LossLeaP can autonomously learn a much better loss, which results in a reduced cost closer to the Oracle one.

4.6.3.2 *In-depth analysis of the learned loss function*

The right plot of Figure 67 offers a glance at the fairly complex loss function captured by the loss-learning DNN of LossLeaP: even if full knowledge of the numerical OPEX models were available, manually devising the shape in the plot would be an exacting task. Our proposed approach effectively automates such design, which lays an important stone in the path to more efficient NI for capacity forecasting and network management in beyond 5G systems.

4.6.3.3 *Conclusions*

Similarly to what observed in Section 4.6.2, loss-learning models can effectively support NI also in the case of resource allocations that target the reduction of OPEX (expressed here as a combination of the costs of end-user QoE disruption and SLA violations). Experiments with real-world measurement data prove the significant advantage that a loss-learning approach yields over a state-of-the-art capacity predictor. These results will also feed the improved design of loss-learning architectures during the second iteration of the project.

4.7 NI to configure a Reconfigurable Intelligent Surface

Evaluation E7 targets on the assessment of NI solutions for the control of Reconfigurable Intelligent Surfaces (RIS). The DAEMON consortium performed assessments of challenges and solutions related to E7 via activity A23. **Table 25** summarizes the tools, KPIs, TRL, PoC plans, approximate progress and main innovations of such activity.

Table 25. *List of activities for E7.*

ID	Name	Evaluation	Tool	Planned KPIs	Collected KPIs	Target TRL	Planned for PoC demo	Progress
A23	Reconfigurable Intelligent Surfaces Prototype	**E7**	T9	K5, K6	None	3	Yes	10%
	Main innovation: Increase spectrum capacity by using reconfigurable reflectors							

A single activity is devoted to the evaluation of NI for RIS control. This is due to the fact that this activity was originally included in E1 in the DoA. However, the high specificity of the Beyond Edge domain that RIS create is not aligned with the characteristics of the more traditional vRAN environments targeted in E1: this pushed the introduction of a new and highly focused evaluation E7, dedicated to RIS control only. Although the effort on E7 is expected to be much lower than on other evaluations of more mature network technologies, isolating the activity of RIS allows for more correct and consistent separation of the evaluations across network domains. The current progress of the single activity A23 is detailed next.

4.7.1 Reconfigurable Intelligent Surfaces Prototype (A23)

This activity involves experimental work for the control of RIS. At this stage, the activity has essentially focused on developing testbed T9, which is described in detail in Section 3.1.9. Although no actual evaluation has been carried out yet using T9, the experience matured by implementing the platform itself has proven very useful to understand the characteristics of a real-world RIS system, which will be key during the second iteration of the RIS-related work in the project. Experiment for RIS control will be carried out during such a second iteration and will be reported in the next WP5 deliverable.

5 Conclusion and outlook

In this document, we presented the preliminary results of WP5 work on the evaluation of the performance, sustainability and reliability of the NI-solutions developed in WP3 and WP4. We validated such solutions against 9 target KPIs, which have been measured and assessed by a comprehensive set of 7 evaluations involving technical tools such experimental testbeds, simulators or emulators, and datasets.

In Evaluation E1, we focused on real-time control and non-real-time orchestration of vRAN services & resources. We report three main observations: i) gaps are present in the usage of shared resources across pools of DUs which calls for a redesign of the DU pipeline and for NI-driven approaches to limit the employment of costly and energy-consuming hardware accelerators, which actual implementations will be presented in the next WP5 deliverable; ii) there may be still space for improvement in the absolute performance of the different strategies for traffic classification in the vRAN that we compared, which will be eventually reported in the next WP5 deliverable; iii) specific NI algorithms need to be designed to cope with the power consumption of vBSs, as it is much more complicated than what assumed in literature and it is linked to end-user QoS in intricated ways.

In Evaluation E2, we focused on NI solutions to support network slice management & orchestration operations. Our main contributions are two: i) the development of two solutions to tackle the challenging problem of making VNF placement more energy friendly in complex mobile Edge settings; ii) the development of a complete framework for NI-assisted MANO, putting together a number of components that will be actually implemented and integrated during the second iteration of the project.

In Evaluation E3, we focused on NI solutions that support anomaly detection in real-time in both controlled environments and in a production core network. As a single activity is targeting E3, and due its proximity with Evaluation E5, the consortium is considering the option to merge this evaluation with E5 in the second iteration of the project. This decision will be reflected in the next deliverable of WP5.

In Evaluation E4, we targeted NI solutions for service orchestration and resource allocation algorithms in the Edge micro-domain. Three are the main lines of activities that we report: i) improved, flexible and automated management & orchestration of Edge resources based on NI solutions; ii) study of the impact of NI solutions on the management of specific mobile services in sliced Edge environments; iii) comprehensive comparative assessments of different types of NI models, including based on statistical, control and machine learning tools, for Edge orchestration.

In Evaluation E5, we aimed at evaluating NI solutions for anomaly response. We report two main achievements: i) we demonstrated that traditional Random Forest models perform as a well as more complex neural networks in classification and anomaly detection tasks in highly constrained user-plane environments, and we will build on such insights to implement NI models for line-rate inference in the second iteration of the project; ii) we paved the way for NI-assisted alarm generation (whose integration in a pre-production system will be at the core of the second iteration of the project) based on deep learning approaches that can identify anomalies in the signaling data for roaming operations.

In Evaluation E6, we focused on the evaluation of NI solutions for long-timescale operations, i.e., MANO, VNF placement and the associated resource allocation. We report two main achievements: i) we proved that hybrid NI design combining statistical modelling and ML can outperform pure deep learning approaches in resource allocation tasks; ii) we demonstrated that anticipatory networking tasks can largely benefit from the automation of the design of loss functions for deep learning models.

In Evaluation E7, we targeted the assessment of NI solutions for the control of Reconfigurable Intelligent Surfaces (RIS). Although no actual evaluation has been yet carried out in the only activity included in E7, it has been possible to understand the characteristics of a real-world RIS system by implementing the testbed T9. Experiments for RIS control will be carried out during the second iteration of the project.

6 References

[1] D. Bega, M. Gramaglia, A. Banchs, V. Sciancalepore, K. Samdanis, and X. Costa-Perez, 'Optimising 5G infrastructure markets: The business of network slicing', in IEEE INFOCOM 2017 - IEEE Conference on Computer Communications, May 2017, pp. 1–9, doi: 10.1109/INFOCOM.2017.8057045.

[2] M. Masoudi et al., 'Green Mobile Networks for 5G and Beyond', IEEE Access, vol. 7, pp. 107270–107299, 2019, doi: 10.1109/ACCESS.2019.2932777.

[3] E. Ahvar, A.-C. Orgerie, and A. Lébre, 'Estimating Energy Consumption of Cloud, Fog and Edge Computing Infrastructures', IEEE Trans. Sustain. Comput., pp. 1–1, 2019, doi: 10.1109/TSUSC.2019.2905900.

[4] J.-M. Horcas, M. Pinto, and L. Fuentes, 'Context-aware energy-efficient applications for cyber-physical systems', Ad Hoc Netw., vol. 82, pp. 15–30, Jan. 2019, doi: 10.1016/j.adhoc.2018.08.004.

[5] D. Bega, A. Banchs, M. Gramaglia, X. Costa-Perez, and P. Rost, 'CARES: Computation-Aware Scheduling in Virtualized Radio Access Networks', IEEE Trans. Wirel. Commun., vol. 17, no. 12, pp. 7993–8006, Dec. 2018, doi: 10.1109/TWC.2018.2873324.

[6] J. A. Ayala-Romero, A. Garcia-Saavedra, M. Gramaglia, X. Costa-Perez, A. Banchs, and J. J. Alcaraz, 'vrAIn: A Deep Learning Approach Tailoring Computing and Radio Resources in Virtualized RANs', in The 25th Annual International Conference on Mobile Computing and Networking, Los Cabos, Mexico, Oct. 2019, pp. 1–16, doi: 10.1145/3300061.3345431.

[7] I. Yoo, M. F. Imani, T. Sleasman, H. D. Pfister, and D. R. Smith, 'Enhancing Capacity of Spatial Multiplexing Systems Using Reconfigurable Cavity-Backed Metasurface Antennas in Clustered MIMO Channels', IEEE Trans. Commun., vol. 67, no. 2, pp. 1070–1084, Feb. 2019, doi: 10.1109/TCOMM.2018.2876899.

[8] D. Bega, M. Gramaglia, M. Fiore, A. Banchs, and X. Costa-Perez, 'DeepCog: Cognitive Network Management in Sliced 5G Networks with Deep Learning', Apr. 2019.

[9] D. Bega, M. Gramaglia, M. Fiore, A. Banchs, and X. Costa-Perez, 'AZTEC: Anticipatory Capacity Allocation for ZeroTouch Network Slicing', presented at the IEEE INFOCOM 2020 - IEEE Conference on Computer Communications, Apr. 2019.

[10] I. Yoo, M. F. Imani, T. Sleasman, H. D. Pfister, and D. R. Smith, 'Enhancing Capacity of Spatial Multiplexing Systems Using Reconfigurable Cavity-Backed Metasurface Antennas in Clustered MIMO Channels', IEEE Trans. Commun., vol. 67, no. 2, pp. 1070–1084, Feb. 2019, doi: 10.1109/TCOMM.2018.2876899.

[11] S. Akcay, A. Atapour-Abarghouei, and T. P. Breckon, 'GANomaly: Semi-Supervised Anomaly Detection via Adversarial Training', ArXiv180506725 Cs, Nov. 2018, Accessed: Jun. 09, 2020. [Online]. Available: http://arxiv.org/abs/1805.06725.

[12] D. Bega, M. Gramaglia, M. Fiore, A. Banchs, and X. Costa-Perez, 'DeepCog: Optimizing Resource Provisioning in Network Slicing With AI-Based Capacity Forecasting', IEEE J. Sel. Areas Commun., vol. 38, no. 2, pp. 361–376, Feb. 2020, doi: 10.1109/JSAC.2019.2959245.

[13] Rodolphe Legouable. (2021). D2.3 Final 5G-EVE end to end facility description (1.0). Zenodo. https://doi.org/10.5281/zenodo.5070253.

[14] Tsung-Yi Lin et al. 2015. Microsoft COCO: Common Objects in Context. arXiv:1405.0312.

[15] P. Mursia, V. Sciancalepore, A. Garcia-Saavedra, L. Cottatellucci, X. C. Pérez and D. Gesbert, "RISMA: Reconfigurable Intelligent Surfaces Enabling Beamforming for IoT Massive Access," in *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 4, pp. 1072-1085, April 2021, doi: 10.1109/JSAC.2020.3018829.

[16] T. Song et al., "Performance evaluation of integrated smart energy solutions through large-scale simulations," in 2011 IEEE International Conference on Smart Grid Communications (SmartGridComm). IEEE, 2011, pp. 37–42 https://github.com/Olivier-Boudeville-EDF/Sim-Diasca.

[17] https://openairinterface.org/.

[18] https://p4.org/.

[19] C. Sonmez, A. Ozgovde, and C. Ersoy, "Edgecloudsim: An environment for performance evaluation of edge computing systems," *Transactions on Emerging Telecommunications Tech- nologies*, vol. 29, no. 11, p. e3493. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/ett.3493.

[20] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," IEEE Transactions on Vehicular Technology, vol. 68, no. 1, pp. 856–868, Jan 2019.

[21] EuropeanUnion.2016.EUGeneralDataProtectionRegulation(GDPR):Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data ProtectionRegulation). RetrievedOctober18,2021fromhttps://gdpr-info.eu/.

[22] J. A. Ayala-Romero, A. Garcia-Saavedra, X. Costa-Perez and G. Iosifidis, "Bayesian Online Learning for Energy-Aware Resource Orchestration in Virtualized RANs," IEEE INFOCOM 2021 - IEEE Conference on Computer Communications, 2021, pp. 1-10, doi: 10.1109/INFOCOM42981.2021.9488845.

[23] Francesc Wilhelmi. (2020). [ITU-T AI Challenge] Input/Output of project "Improving the capacity of IEEE 802.11 WLANs through Machine Learning" [Data set]. Zenodo. https://doi.org/10.5281/zenodo.4106127.
[24] Barrachina-Munoz, S., Wilhelmi, F., Selinis, I., & Bellalta, B. (2019, April). Komondor: A wireless network simulator for next-generation high-density WLANs. In 2019 Wireless Days (WD) (pp. 1-8). IEEE.
[25] https://www.itu.int/en/ITU-T/AI/challenge/2020/Pages/default.aspx.
[26] IEEE. TGax Simulation Scenarios. Doc.: IEEE 802.11-14/0980r16. 2015. Available online: https://mentor.ieee.org/802.11/dcn/14/11-14-0980-16-00ax-simulation-scenarios.docx.
[27] Adame, T., Carrascosa, M., Bellalta, B. The TMB path loss model for 5 GHz indoor WiFi scenarios: On the empirical relationship between RSSI, MCS, and spatial streams. In Proceedings of the 2019 Wireless Days (WD), Manchester, UK, 24–26 April 2019; pp. 1–8.
[28] Barrachina-Muñoz, S.; Wilhelmi, F.; Bellalta, B. Dynamic Channel Bonding in Spatially Distributed High-Density WLANs. IEEE Trans. Mob. Comput. 2020, 19, 821–835.
[29] I. Sharafaldin, A. Habibi Lashkari and Ali A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization", 4th International Conference on Information Systems Security and Privacy (ICISSP), Portugal, January 2018 .
[30] M. Zink, K. Suh, Y. Gu, and J. Kurose, "Characteristics of YouTube network traffic at a campus network - Measurements, models, and implications," Elsevier Comput. Netw., vol. 53, no. 4, pp. 501–514, Mar. 2009.
[31] W. Yu, F. Liang, X. He, W. G. Hatcher, C. Lu, J. Lin, X. Yang, A survey on the edge computing for the internet of things, IEEE Access 6 (2018)6900–6919. doi:10.1109/ACCESS.2017.2778504.
[32] Phoronix Test Suite details: https://openbenchmarking.org/tests/pts.
[33] GEC Numerical variability model download as Clafer model: https://hadas.caosd.lcc.uma.es/edgenvmfgcs.txt.
[34] M. Gramaglia et al., "Initial design of real-time control and VNF intelligence mechanisms", DAEMON deliverable D3.1.
[35] Samsung. 2019. Virtualized Radio Access Network: Architecture, Key technologies and Benefits. Technical Report (2019).
[36] Rethink Technology Research. 2020. Special Report: Open Networks. Technical Report (2020).
[37] NGMN Alliance. 2019. 5G E2E Technology to Support Verticals URLLC Requirements.
[38] F. Kaltenberger, A. P. Silva, A. Gosain, L. Wang, and T. Nguyen. 2020. OpenAirInterface: Democratizing innovation in the 5G Era. Computer Networks (2020), 107284.
[39] J. Ding, R. Doost-Mohammady, A. Kalia, and L. Zhong. 2020. Agora: Real-time massive MIMO baseband processing in software. In Proceedings of ACM CoNEXT '20. ACM.
[40] Samsung. 2019. Virtualized Radio Access Network: Architecture, Key technologies and Benefits. Technical Report (2019).
[41] Wang Tsu Han and Raymond Knopp. 2018. OpenAirInterface: A pipeline structure for 5G. In 2018 IEEE 23rd International Conference on Digital Signal Processing (DSP). IEEE, 1–4.
[42] J. A. Ayala-Romero, A. Garcia-Saavedra, M. Gramaglia, X. Costa Perez, A. Banchs, and J. J. Alcaraz. 2020. vrAIn: Deep Learning based Orchestration for Computing and Radio Resources in vRANs. IEEE Transactions on Mobile Computing (2020), 1–1. https://doi.org/10.1109/TMC.2020.3043100.
[43] I. Gomez-Miguelez, A. Garcia-Saavedra, P. D. Sutton, P. Serrano, C. Cano, and D. J Leith. 2016. srsLTE: an open-source platform for LTE evolution and experimentation. In Proceedings of the Tenth ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation, and Characterization. 25–32.
[44] Y. Sun and J. R Cavallaro. 2011. A flexible LDPC/turbo decoder architecture. Journal of Signal Processing Systems 64, 1 (2011), 1–16.
[45] O-RAN Alliance. 2020. Cloud Architecture and Deployment Scenarios for O-RAN Virtualized RAN v02.01 (O-RAN.WG6.CAD-v02.01). Technical Report.
[46] D. De Vleeschauwer et al., "Initial design of intelligent orchestration and management mechanisms", DAEMON deliverable D4.1.
[47] G. Garcia-Aviles, A. Garcia-Saavedra, M. Gramaglia, X. Costa-Perez, P. Serrano, A. Banchs. Nuberu: Reliable RAN Virtualization in Shared Platforms. In *ACM MobiCom*, 2021.
[48] N. Siegmund, A. Grebhahn, S. Apel, and C. Kästner. 2015. Performance-Influence Models for Highly Configurable Systems. In Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering (Bergamo, Italy) (ESEC/FSE 2015). Association for Computing Machinery, New York, NY, USA, 284–294. https://doi.org/10.1145/2786805.2786845.
[49] Martin Bauer. 2019. A Comparison of Six Constraint Solvers for Variability Analysis. Technical Report. University of Passau.
[50] C. Sundermann, T. Thüm, I. Schaefer, Evaluating #sat solvers on industrial feature models, in: Proceedings of the 14th Int. Conference on Variability Modelling of Software-Intensive Systems, VAMOS '20, ACM, New York, NY, USA, 2020.
[51] M. Amor and L. Fuentes, "Energy-efficient Deployment of IoT Applications in Edge-based Infrastructures: A Software Product Line Approach," in IEEE Internet of Things Journal, doi: 10.1109/JIOT.2020.3030197.

[52] Y. Liu, X. Yuan, Z. Xiong, J. Kang, X. Wang and D. Niyato, "Federated Learning for 6G Communications: Challenges, Methods, and Future Directions," China Communications, Special Issue: "6G Mobile networks: Emerging technologies and applicaitons, September 2020.
[53] S. Telecom. (2018) The telecom dataset. [Online]. Available: http://sguangwang.com/TelecomDataset.html.
[54] K. Djemame and A. Aljulayfi, "A machine learning based context-aware prediction framework for edge computing environments," pp. 143–150, April 2021. [Online]. Available: https://eprints.whiterose.ac.uk/173188/.
[55] ETSI, "Network Functions Virtualisation (NFV); Management and Orchestration," ETSI, Specification, 2014. [Online]. Available: https://www.etsi.org.
[56] P. Soto, D. De Vleeschauwer, M. Camelo, et. al., "Towards Autonomous VNF Auto-scaling using Deep Reinforcement Learning," 2021 Eight International Conference on Software Defined Systems (SDS), to be published.
[57] A. Raffin et al., "Stable baselines3," https://github.com/DLR-RM/stable-baselines3, 2019.
[58] C. Busse-Grawitz, R. Meier, A. Dietmüller, T. Bühler, and L. Vanbever. 2019. pForest: In-Network Inference with Random Forests. CoRR abs/1909.05680 (2019). arXiv:1909.05680 http://arxiv.org/abs/1909.05680.
[59] C. Zheng and N. Zilberman. 2021. Planter: Seeding Trees within Switches. Association for Computing Machinery, New York, NY, USA, 12–14. https://doi.org/10.1145/3472716.3472846.
[60] D. Bega, M. Gramaglia, M. Fiore, A. Banchs, and X. Costa-Perez, "Deepcog: Cognitive network management in sliced 5g networks with deep learning," in IEEE INFOCOM 2019 - IEEE Conference on Com- puter Communications, 2019, pp. 280–288.
[61] J. Marquez-Barja et al., "Smart Highway : ITS-G5 and C2VX based testbed for vehicular communications in real environments enhanced by edge/cloud technologies," in 2019 European Conference on Networks and Communications (EuCNC), Abstracts, Valencia, Spain, 2019.
[62] Jong-Hyouk Lee and Kamal Preet Singh. 2020. SwitchTree: in-network computing and traffic analyses with Random Forests. Neural Computing and Applications (2020), 1–12.
[63] Zhaoqi Xiong and Noa Zilberman. 2019. Do Switches Dream of Machine Learning? Toward In-Network Classification. In Proceedings of the 18th ACM Workshop on Hot Topics in Networks (Princeton, NJ, USA) (HotNets '19). Association for Computing Machinery, New York, NY, USA, 25–33. https://doi.org/10.1145/3365609.3365864.
[64] Giuseppe Siracusano and Roberto Bifulco. 2018. In-network Neural Networks. CoRR abs/1801.05731 (2018). arXiv:1801.05731 http://arxiv. org/abs/1801.05731.
[65] Giuseppe Siracusano, Salvator Galea, Davide Sanvito, Mohammad Malekzadeh, Hamed Haddadi, Gianni Antichi, and Roberto Bifulco. 2022. Re-architecting Traffic Analysis with Neural Network Interface Cards. In 19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22). USENIX Association, Renton, WA. https: //www.usenix.org/conference/nsdi22/presentation/siracusano.
[66] Muhammad Shahbaz Ishan Gaur Tushar Swamy, Alexander Rucker and Kunle Olukotun. 2022. Taurus: A Data Plane Architecture for Per-Packet ML. Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (2022).
[67] Arunan Sivanathan, Hassan Habibi Gharakheili, Franco Loi, Adam Rad ford, Chamith Wijenayake, Arun Vishwanath, and Vijay Sivaraman. 2019. Classifying IoT Devices in Smart Environments Using Network Traffic Characteristics. IEEE Transactions on Mobile Computing 18, 8 (2019), 1745–1759. https://doi.org/10.1109/TMC.2018.2866249.
[68] Maurizio Dusi, Manuel Crotti, Francesco Gringoli, and Luca Salgarelli. 2008. Detection of Encrypted Tunnels Across Network Boundaries. 2008 IEEE International Conference on Communications (2008), 1738– 1744. http://netweb.ing.unibs.it/~ntw/tools/traces/.
[69] Alice Este, Francesco Gringoli, and Luca Salgarelli. 2011. On-line SVM traffic classification. In 2011 7th International Wireless Communications and Mobile Computing Conference. 1778–1783. https://doi.org/10.1109/ IWCMC.2011.5982804.
[70] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A. Ghorbani. 2009. A detailed analysis of the KDD CUP 99 data set. In 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications. 1–6. https://doi.org/10.1109/CISDA.2009.5356528.
[71] Nour Moustafa and Jill Slay. 2015. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In 2015 Military Communications and Information Systems Conference (MilCIS). 1–6. https://doi.org/10.1109/MilCIS.2015.7348942.
[72] Nour Moustafa and Jill Slay. 2016. The Evaluation of Network Anomaly Detection Systems: Statistical Analysis of the UNSW-NB15 Data Set and the Comparison with the KDD99 Data Set. Inf. Sec. J.: A Global Perspective 25, 1–3 (apr 2016), 18–31. https://doi.org/10.1080/19393555. 2015.1125974.