H2020 DAEMON Project
Grant Agreement No. 101017109

Deliverable 4.1

Initial design of intelligent orchestration and management mechanisms

Abstract

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

Document properties

Document number	D4.1
Document title	Initial design of intelligent orchestration and management mechanisms
Document responsible	Georgios Iosifidis (TUD)
Document editor	Danny De Vleeschauwer (NBL), Chia-Yu Chang (NBL)
Editorial team	Danny De Vleeschauwer (NBL) Chia-Yu Chang (NBL) Marco Fiore (IMDEA) Sergi Alcalá (IMDEA) Andres Garcia Saavedra (NEC) Gines Garcia Aviles (i2CAT) Ivan Paez (ADL) Gabriele Baldoni (ADL) Andra Lutu (TID) Miguel Camelo (IMEC) Nina Slamnik-Krijestorac (IMEC) Paola Soto-Arenas (IMEC) Mercedes Amor (UMA) Angel Cañete (UMA) Lidia Fuentes (UMA) Panagiotis Demestichas (WINGS) Aspa Skalidi (WINGS) Evangelos Kosmatos (WINGS) Ioannis-Prodromos Belikaidis (WINGS) Andreas Georgakopoulos (WINGS) Alexandros Kostopoulos (OTE) Georgios Iosifidis (TUD)
Target dissemination level	Public
Status of the document	Final
Version	1.0

Production properties

Reviewers	Marco Gramaglia (UC3M)

Document history

Revision	Date	Issued by	Description
0.1	18/10/21	All partners	Initial content
0.2	25/10/21	Editor, WP4 lead	First version, ready to be reviewed
0.3	1/11/21	All partners	Updated content based on feedback of editor and WP4 lead
0.4	7/11/21	Marco Gramaglia	Version reviewed by Marco Gramaglia
0.5	17/11/21	All partners	Updated content based on feedback from Marco Gramaglia
0.6	19/11/21	Editor	Review by editor
0.7	22/11/21	WP4 lead	Review by WP4 lead
0.8	26/11/21	Project lead	Review by project lead
1.0	30/11/21	Project lead	Final version sent to SygMa

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

2G – Second Generation
3G – Third Generation
4G – Fourth Generation
5G – Fifth Generation
6G – Sixth Generation
AF –Application Function
ADAM – ADAptive Moment estimation
AI – Artificial Intelligence
API – Application Programming Interface
AR – Augmented reality
B5G – Beyond 5th Generation
BBU – Baseband Unit
BS –Base Station
CNF – Cloud-native Network Function
CNN – Convolutional Neural Network
CPS – Cyber Physical System
CPU – Central Processing Unit
COTS – Commercial Off The Shelf
CU – Central Unit
CQI – Channel Quality Indicator
CRAN – Cloud Radio Access Network
DL – DownLink
DNN – Deep Neural Network
DNS – Domain Name Service
DQN – Deep Q-Network
DRA – Diameter Routing Agent
DRL – Deep Reinforcement Learning
DU – Distributed Unit
eNB – evolved Node B
ES – Exponential Smoothing
ETSI - European Telecommunications Standards Institute
FCNN – Fully Connected Neural Network
FIFO – First In, First Out
FL – Federated Learning
GMM – Gaussian Mixture Model
gNB – next generation Node B
GPU – Graphical Processing Unit
GTP – General Packet Radio Service (GPRS) Tunneling Protocol
HDD – Hard Disk Drive
IMSI – International Mobile Subscriber Identity
IP – Internet Protocol
IPX – Internet Protocol eXchange
IMEI – International Mobile Equipment Identity
IoT – Internet of Things
IoU – Intersection over Union
KL – Kullback–Leibler
KPI – Key Performance Indicator
LA – Learning Agent
LFU – Least Frequently Used
LRU – Least Recently Used
LSTM – Long Short-Term Memory
LTE – Long Term Evolution

M2M – Machine To Machine
MANO – MANagement and Orchestration
mAP – mean Average Precision
MAP – Mobile Application Part
MAPE – Monitor, Analyze, Plan and Execute
MBS – Macro Base Station
MCF – Multi-Commodity Flow
MCS – Modulation and Coding Scheme
MDP – Markovian Decision Process
MEC – Multi-access Edge Computing
MILP – Mixed Integer Linear Programming
MNO – Mobile Network Operator
MSE – Mean Square Error
MVA – Mobile Video Analytics
MDP – Markov Decision Process
ML – Machine Learning
MLP – MultiLayer Perceptron
MOS – Mean Opinion Score
NF – Network Function
NFV – Network Function Virtualization
NFVI – Network Function Virtualization Infrastructure
NI – Network Intelligence
NN – Neural Network
NRT – Near Real Time
NS – Network Service
NSSI –Network Slice Subnet Instance
NWDAF – NetWork Data Analytics Function
OCO – Online Convex Optimization
OPEX – OPerating EXpenses
OS – Operating System
OSM – Open Source MANO
OSFPF – Optimal Service Function Placement Framework
OTAF – Optimal Task Assignment Framework
PDP – Packet Data Protocol
PNF – Physical Network Function
PoE – Power-over-Ethernet
POMDP – Partially Observable Markov Decision Process
PRB – Physical Resource Blocks
PUSCH – Physical Uplink Shared Channel
PUCCH – Physical Uplink Control Channel
QoS – Quality of Service
QoE – Quality of Experience
RAM – Random Access Memory
RAN – Radio Access Network
RAPL – Running Average Power Limit
RCS – Rich Communication Services
REC – Reduction of Energy Consumption
ReLU – Rectified Linear Unit
RNN – Recurrent Neural Network
RF – Radio Frequency
RIC – RAN Intelligence Controller
RL – Reinforcement Learning
ROS – Robot Operating System

RT – Real Time
RTT – Round Trip Time
RU – Radio Unit
SAE - Stacked AutoEncoder
SBS – Small Base Station
SCCP – Signaling Connection Control Part
SDN – Software Defined Networking
SDR – Software Defined Radio
SF – Small Factor
SFC – Service Function Chain
SIM – Subscriber identity Module
SIP – Session Initiation Protocol
SLA – Service Level Agreement
SLO – Service Level Objective
SMO – Service Management and Orchestration
SMT – Satisfiability Modulo Theories
SNR – Signal to Noise Ratio
SS7 – Signaling System 7
STP – Signaling Transfer Point
TAC – Type Allocation Code
TES – Thresholded Exponential Smoothing
TIS – Tasks Implementation Selector
TPC – Transmission Power Control
TTL – Time-To-Live
UCI – Uplink Control Information
UE – User Equipment
UL – UpLink
URLLC – Ultra-Reliable Low-Latency Communications
VAE – Variational Autoencoder
vBS – virtualized Base Station
VIM – Virtual Infrastructure Manager
VM – Virtual Machine
VMNO – Virtual Mobile Network Operator
VN – Virtual Network
VNF – Virtual Network Function
VNFM – Virtual Network Function Manager
VoIP – Voice over IP
VR – Virtual Reality
vRAN – virtualized Radio Access Network
WLAN – Wireless Local Area Network
YAML – Yet Another Markup Language

Executive summary

This deliverable presents the progress made by DAEMON on the design and development of NI-assisted functionalities for Beyond fifth Generation (B5G) in the area of service and resource management and orchestration. This overall goal is tackled by considering four specific B5G functionalities: (i) energy-aware Virtual Network Function (VNF) orchestration, (ii) capacity forecasting, (iii) automated anomaly detection, and (iv) self-learning management and orchestration.

Section 2 discusses how these building blocks are related and how they can improve the KPIs identified at the start of the project. The rest of the sections zoom in on each of the building blocks.

Energy-aware VNF orchestration focuses on problems concerning orchestration and lifecycle management of VNFs, Cloud-native Network Functions (CNFs) and policy (re-)configuration of Physical Network Functions (PNFs), towards achieving a balance between performance and energy usage. This implies trading off performance for battery lifetime when running Network Functions (NFs) on untethered devices (e.g., sensors, mobile or solar-powered base stations, etc.), but also understanding the long-term energy toll associated with the whole system. Moreover, while this activity has a clear target on the energy consumption of VNFs, it also considers the energy footprint inherent to the Network Intelligence (NI) governing the lifecycle of such NFs. The main highlights obtained in this reporting period are:

- Two approaches for the orchestration of virtualized services are proposed at two different levels: virtualizing NFs at user/data level and virtualizing the Radio Access Network functions.
- We present a novel process for decomposing VNFs representing their dependencies and constraints with a service-call graph model, which is used to assess the energy consumption at level of VNF code. We propose a first version of an energy-aware placement algorithm based on mathematical models and a proof of concept that shows it may be a good starting point.

Capacity forecasting predicts the capacity of computing and cloud, spectrum, and networking resources, and infers the model governing diverse data-generating processes such as mobility patterns. The algorithms devised in this task need to take into account all possible sources of information for performing the forecasting and inference legwork, with focus on the monetary implications that prediction and recommendations may have on the system, as well as the desired level of reliability. The research carried out by the project in capacity forecasting includes the following highlights:

- We introduce the very first hybrid prediction model for mobile networks, which integrates and jointly optimizes statistical modelling and machine learning methods. The sizeable performance improvement that this design attains over state-of-the-art deep neural network architectures in two practical NI use cases indicates that hybrid approaches may set a new standard for forecasting in mobile network environments.
- We propose a novel automated approach that learns to predict resource requirements in a way to flexibly optimize tangled performance objectives. Our original design allows discovering complex relationships between the objective and the model parameters, and then use it as a highly tailored loss function to train a dedicated predictor, improving forecasting quality.

Automated anomaly response is studied in the framework of the massive-density of Internet of Things (IoT) small devices, dense virtual radio access points, etc., that will integrate next-generation systems making B5G particularly vulnerable to anomalous behavior. Anomalies may have different natures (non-malicious anomalies caused by faulty equipment or misconfigurations versus malicious, anomalies caused by attackers), different scopes (data plane anomalies versus control/management plane anomalies), and even different scales (long-term anomalies versus short-term anomalies). Specific NI solutions are being investigated that constitute an effective army of automated tools that detect, analyze and act against such anomalous behavior. The main highlights obtained in this reporting period are as follows:

- A novel approach towards federated learning for anomaly detection.
- Anomaly detection in the case of an operational roaming platform, where we expose the problem, explore the available dataset and showcase initial results towards building an automated approach for tackling anomalies within this complex system.

Finally, self-learning MANagement and Orchestration (MANO) handles problems related to autonomous orchestration and management with minimal human intervention. This includes algorithms that dynamically adapt their operation online and the interfaces that enable the gathering of monitoring information that allows adjusting the parameters of the NI deployed in the system. The innovations go beyond deploying decision-making engines based on, e.g., Reinforcement Learning (RL): mechanisms that enable NI to self-learn its own policies, steered by high-level Quality of Experience (QoE) targets and business KPIs – instead of strict Quality of Service (QoS) goals and technical KPIs – are being studied and implemented. Besides the autonomous operation of the algorithms, this task also considers aspects

related to the explainability and trustworthiness of the NI algorithms, providing human operators with the means to track down and interpret the action trajectory of autonomous decision-making engines, e.g., to perform root-cause analysis and troubleshooting. In the first period of the project that is covered at this deliverable the highlights are:

- A mathematical formulation of the resource allocation problem when a network service is requested, with an argumentation that Machine Learning (ML) is beneficial in this context because requirements of requests can be learned as more requests are handled. Some initial results indicate the potential of this approach.
- Two resource scaling use case, where, throughout the lifetime of the network service, the allocated resources follow the demand fluctuations induces by the time-varying behavior of the users of the target network service.

1 Introduction

WP4's main objective is to design NI-based solutions for the functionality related to service and resource orchestration and management, particularly covering both DAEMON objective 2 (Developing specialized NI-assisted network functionalities for B5G systems.) and several specific Key Performance Indicators (KPIs) of objective 4 (Demonstrating the viability and performance of NI-native B5G networks). In this deliverable, we provide the initial design of these NI solutions with preliminary results and first performance evaluations, while the refined design and final design of these NI solutions will be presented in the two future WP4 deliverables, respectively.

To this aim, NI-based solutions are investigated on top of the DAEMON framework, providing NI-assisted functionality for the management and orchestration in the B5G system to attain a variety of goals: performance, efficiency, sustainability, and reliability. In essence, the activities in WP4 will be based on the architecture defined in WP2 to not only ensure the developed NI solutions are aligned with the specific needs of mobile network systems, but also to provide NI models for a comprehensive NI management environment along different micro-domains. Also, WP4 provides feedback to WP2 to refine the requirements defined in and the framework designed in WP2. To this end, every NI solution devised in WP4 will be continuously evaluated during the lifetime of DAEMON according to the periodic interaction with WP2 tasks. Moreover, NI solutions developed in WP4 will be extensively examined and evaluated over the available simulation/emulation platforms, testbed experiment, and large-scale datasets to observe the algorithm efficiency in WP5. In this sense, the gap between algorithm design and system development can be reduced significantly and the NI-based solution can be further enhanced after considering real-world implementation limitations. Finally, even though WP3 and WP4 are targeting to design NI solutions with different time granularities and for different micro-domains, these two WPs are both investigating potential NI solutions for the B5G system and therefore their generated knowledge (belonging to different timescales and domains) shall be handled properly by WP2 to make the cooperation and coordination possible in an adaptive and self-learning mobile network. A plot depicting these interactions between WP4 and other related WPs can be found in Figure 1.***2.1.1***.

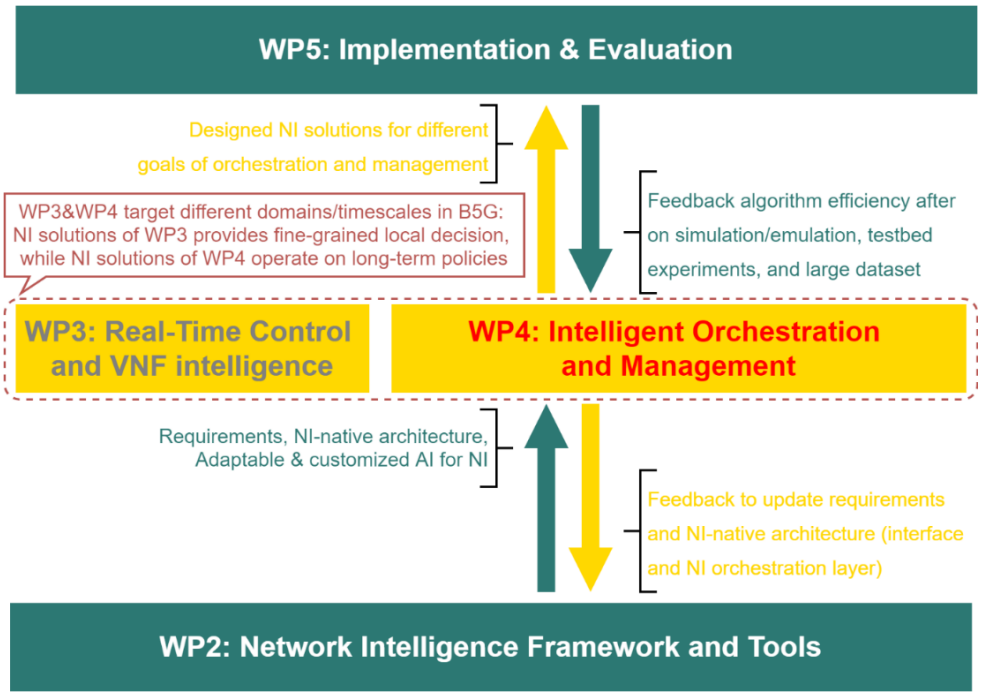


Figure 1.2.1.1. *Relationship between WP4 and other WPs.*

This deliverable is organized as follows. First, in Section 2, it discusses the four WP4 goals of DAEMON to design an intelligent orchestration and management solution, and how each of these four goals can provide their outputs to others to further refine the designed NI solutions. Such interaction can be fed back to WP2 to draft the interfaces between NI instances, and we also provide a graph modeling approach in Section 2 for the same purpose. Moreover, these four distinct goals are respectively related to the four Tasks of WP4. Therefore, from Section 3 to 6, we elaborate on 13 NI solutions grouped according to WP4 goal. To be more specific, three NI solutions are provided in Section 3 to take energy-awareness into account when orchestrating VNFs over different domains. In Section 4, there are four NI solutions providing the necessary forecasting functionalities for network state and resource utilization to assist the decision-making of network functionalities. Then, in Section 5, the anomaly detection is targeted

by our two proposed NI solutions to automatically detect unexpected emerging behaviors in the network and trigger relevant signals *to* reduce system reaction time. Finally, four solutions are designed in Section 6 to enable the self-learning management and orchestration operations, e.g., placement and scaling, for NFs and network services (NSs). The conclusion and outlook are drawn in Section 7. In addition, to provide substantial background knowledge from prior arts, a thorough state-of-the-art review on each NI solution is summarized in the appendix Section 8.

2 Functional Architecture

DAEMON NI native architecture described in D2.1 [1] paves the way to establish an ecosystem for accommodating NI innovations into an end-to-end 5G mobile network. In particular, several micro-domains are identified to realize the orchestration and control of targeting NFs, e.g., VNFs and CNFs, to increase management granularity among distinct orchestrators and controllers. Note that this NI-native architecture aims to reconstruct the legacy decision-making scheme from centralized side-loaded intelligence applications into a deep intelligence integration spanning several network micro-domains and operating in different time scales. Therefore, NI solutions can closely exchange data information from the control and data plane NFs, and hence, promptly react for decision making processes to fit in network real-time requirements. This section introduces first the goals of NI-based network orchestration solutions that are discussed in later sections. Then, it presents a general dataflow modeling method which can be applied to different domains, and NI solution can be decomposed into graph of components, in which NI algorithms acts as operators to compute data.

2.1 Network intelligence goals for orchestration and management

In this deliverable, our focus is on the management and orchestration perspective within the DAEMON NI-native architecture, in which several NI-based solutions are investigated to facilitate the management of NSs and NFs during their lifecycle, so as to reach four identified goals: (a) Energy-aware orchestration and placement, (b) Network state and capacity forecasting, (c) Automated anomaly detection, and (d) Self-learning MANO. These goals can be mapped to the identified NI use cases defined in D2.1 [1], targeting controller and orchestrator, targeting micro-domain(s), and the targeting DAEMON KPIs, as summarized in Table 1. Furthermore, for each goal, several NI-based solutions are presented in this deliverable as shown in Figure **2.1.1**, together with their interactions. In the following paragraphs, we give a compact introduction to each NI-based solution in terms of its challenges, objective, constraints, and potential interactions towards other goals for intelligence orchestration and management.

Table 1. *Goals for intelligence orchestration and management*

Goal	Mapped use case in D2.1	Targeting controller /orchestrator	Targeting micro-domain(s)	Targeting DAEMON Network KPIs
Energy-aware orchestration and placement	EAWVNF	Non-Real-Time RAN controller, Edge/Fog Orchestrator	Edge, Core	(K1) VNF energy consumption reduction, (K2) Saving of computational resources at the edge,
Network state and capacity forecasting	CFORE	Edge/Fog Orchestrator	Edge, Transport, Core	(K2) Saving of computational resources at the edge, (K4) OPEX saving,
Automated anomaly detection	AARES	Non-Real-Time RAN controller	Transport, Core	(K5) Reliability, (K7) Anomaly detection recall and sensitivity, (K8) Vertical service response time
Self-learning management and orchestration	SLMANO	Edge/Fog orchestrator, End-to-end orchestrator	Core	(K4) OPEX saving, (K5) Reliability, (K8) Vertical service response time, (K9) Optimality gap of network management decisions

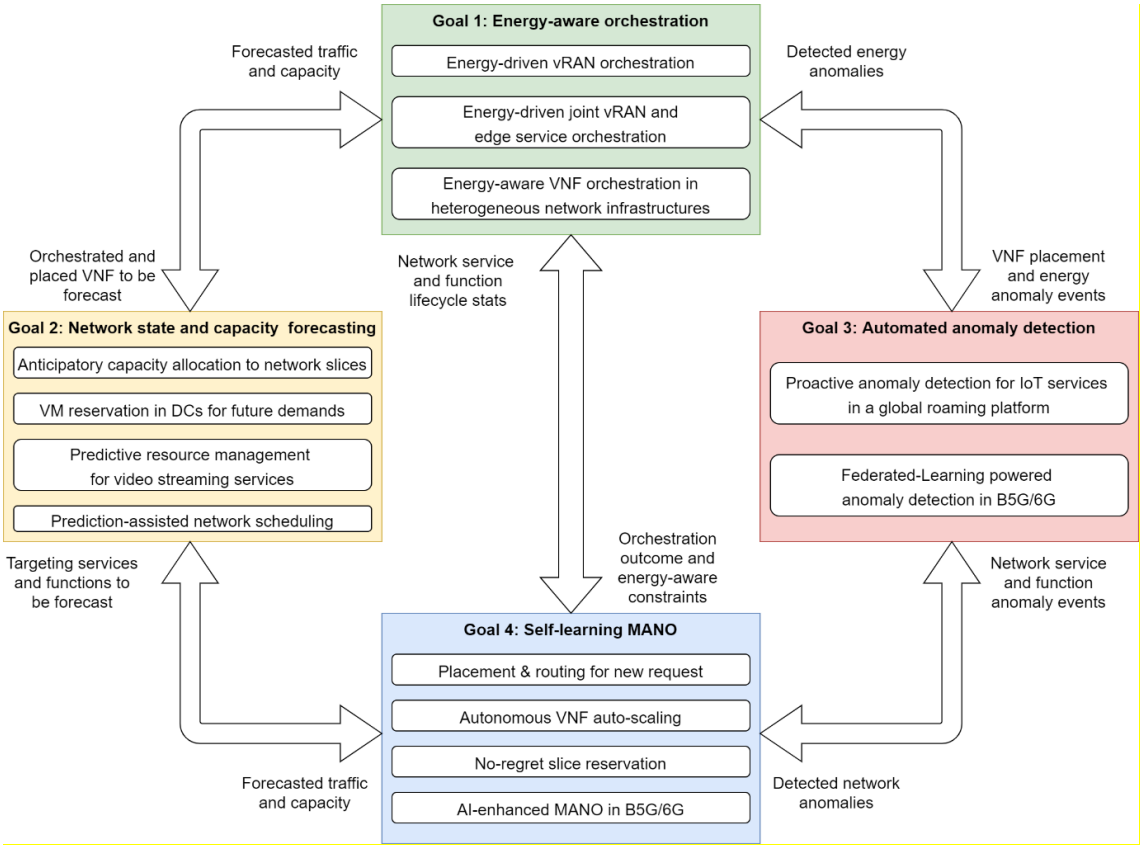


Figure 2.1.1. *Four identified goals for intelligent orchestration and management.*

2.1.1 Goal 1: Energy-aware orchestration

The work that has been carried out in the first 11 months related to energy aware VNF orchestration has focused on the study of energy consumption at mainly two levels, (i) virtualized Base Stations (vBSs), and (ii) heterogenous network infrastructures. The solutions considered at this point of the project are intended to contribute to the DAEMON KPIs K1 and K2. The proposal that is being developed in the context of Radio Access Networks (RANs), specifically deals with the vBS operation in terms of energy consumption and computational resources' demands for different platforms and scenarios. The aim is to exploit experimentation results to shed light on the relationship between K1 and K2 and operational vBS controls.

On the other hand, an extensible variability model of VNFs and network infrastructures is the core of the algorithms that calculate the optimal solutions of VNF placement in terms of energy consumption, contributing to reach K1. These solutions aim to cope with KPIs K1 and K2 by optimizing VNF orchestration attending to a wider and variable set of network infrastructures, including the Edge. The VNF placement algorithms proposed, for the moment, consider generic code and will be adapted later to specific VNFs with high energy consumption demands.

Goal 1 is closely related to Goal 4, as energy-aware decisions about both VNF orchestration and auto-scaling rely on the same mechanisms to assess the energy footprint of NI in heterogeneous infrastructures.

2.1.2 Goal 2: Network state and capacity forecasting

The forecasting solutions developed in DAEMON during its first 11 months span across multiple network functionalities and include support for (i) anticipatory capacity allocation to network slices, (ii) reservation of Virtual Machine (VMs) in network datacenters for future demands, (iii) predictive management of resources for video streaming services, and (iv) prediction-based scheduling for data analytics. Such an already wide range of solutions will be further extended during the project lifetime and contribute to reaching the DAEMON KPIs K2 and K4: indeed, the NI tools above allow for a more effective allocation of resources in the network edge and core micro-domains and yield substantial savings in terms of required (e.g., compute, memory, transport, or storage) resources and in terms of operating expenses incurred by the Mobile Network Operator (MNO).

The NI solutions aimed at anticipatory resource management are also extremely useful tools that can be leveraged to enhance other NI instances developed in DAEMON, at later stages (e.g., during the second and third "iterations" of the project timeline, in years two and three of its execution). Namely, capacity forecasting models can be combined with approaches for self-learning MANO (see Goal 1 and Goal 4):

the latter typically operate in a responsive way, whereas predictions allow anticipating MANO decisions to accommodate future demands, by building MANO decision-making on top of capacity forecasts. Also, traffic and capacity forecasts are instrumental to a deployment and chaining of VNFs that is efficient from the perspective of network operation and energy consumption: indeed, VNFs rely on different types of network resources, and having information on their availability in advance, allows taking much better-informed decisions in terms of their placement, towards achieving specific operational and performance targets.

2.1.3 Goal 3: Automated anomaly detection

The automated anomaly detection solutions that DAEMON has been developing during these first 11 months revolve around building realistic solutions for real-world operational systems that offer managed connectivity for a massive number of IoT devices. In this context, DAEMON is considering cutting-edge deep learning techniques that will enable the proactive detection of anomalies within these complex systems and that will enable the engineering teams to react faster when anomalies occur (e.g., before the service to the client is severely impacted).

Furthermore, within DAEMON we also consider novel approaches to support anomaly detection, such as Federated Learning (FL). This will add strength to the solutions we build by allowing similar – but separate – entities to collaborate and train powerful anomaly detection methods – each using their own separate datasets, while contributing to building a common stronger approach.

Goal 1 and Goal 3 may benefit from anomaly detection as this can serve as trigger to perform orchestration actions.

2.1.4 Goal 4: Self-learning management and orchestration

Self-learning MANO is one of the key goals of DAEMON to deal with the increasingly complex network management in B5G and Sixth Generation (6G) networks. In these networks determining the best orchestration strategy becomes extremely challenging, because the manual, human-centered approach cannot deal with the increase in problem size and complexity. Moreover, offline pre-trained Machine Learning (ML) models will not react to changing circumstances. Therefore, the self-learning NI targeted in this goal is envisioned to autonomously learn to evaluate the quality of its decision-making to meet both application and network KPIs, while ensuring that operators obtain economic benefits. To be more specific, in this deliverable D4.1, we highlight four different tasks to reach this goal: (i) Placement and routing for new request, (ii) Autonomous VNF auto-scaling, (iii) Virtualized Radio Access Network (vRAN) resources auto-scaling, and (iv) AI-enhanced MANO in B5G/6G.

In addition, the NI-based solutions proposed for these four respective tasks for self-learning MANO will interact with other goals. First, the lifecycle of NSs (e.g., admission control decision) can be provided to Goal 1 in order to produce service- and domain-specific low-energy VNF orchestration. In the opposite direction, self-learning MANO solutions can obtain the results of energy-aware VNF orchestration, and even energy-driven constraints as inputs. In addition, the solutions of Goal 2 can provide its results, such as predicted computing resource, so that the self-learning MANO solution can proactively apply scaling decisions in advance. More specifically, the forecasting solutions will work closely with self-learning MANO to make the orchestration decision-making work in a closed-loop pipeline and fit in the real-time limitations. Finally, the automated anomaly detection solutions of Goal 3 can provide their (potential) detected anomalies to the self-learning MANO for conflict-resolution and re-orchestration operations. In this regard, NS can provide adequate recovery solutions, like scaling in and out or scaling up and down based on economic impact, or even do Service Level Agreement (SLA) negation to verticals to maintain service continuity.

2.2 Unified dataflow deployment

The goal of this threat is to enable self-learning MANO and NI algorithms to manage the increasingly complex networks expected in B5G and 6G. As those networks span end-to-end, it is crucial to enable the deployment of such algorithms across the infrastructure. To this end, we introduce the dataflow programming model which can be used to model distributed NI workflows as those we have in DAEMON.

The dataflow programming model [2] allows to organize programs as directed graphs where nodes are computational units, i.e., the operators, and the arcs between them symbolize the communication channels. Figure **2.2.1** shows a representation of a dataflow graph. Arcs arriving to an operator represent the inputs and those departing from it represents the outputs. Upon receiving its inputs, an operator can trigger its internal computation. Depending on the dataflow model, an operator either waits for all its inputs or for a specific subset of them, referred to as a firing set or a firing rule. Firing rules offer more flexibility to application developers, as they can express more conditions: for example, if the input is optional, the computation can still be triggered when it is missing.

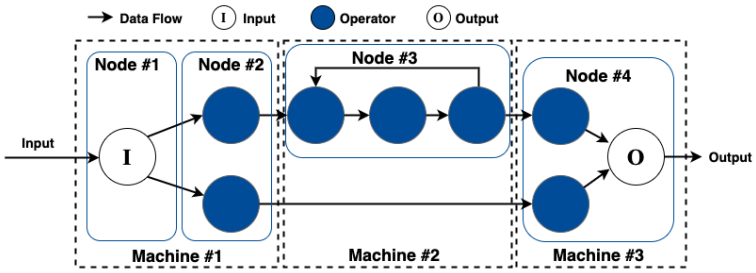


Figure 2.2.1. *Dataflow programing graph.*

In the dataflow programming model applications are composed by three different kinds of components:

- ***Sources***, getting data from outside world,
- ***Sink***, sending data or actions to the outside world, and
- ***Operators***, computing over the data.

Therefore, a dataflow application is composed by a graph of components, including at least a source, an operator and a sink. Developers can focus only on writing and specifying each single component, without needing to worry about how they will be interconnected. This paradigm encourages reutilization of components and therefore code reuse,

The Zenoh Flow[1] framework [3] embraces such a programming model and extends it to the distributed infrastructure ranging from the edge to the cloud. Any NI application could be decomposed as a graph of components, where the NI algorithms act as operators, since they perform computations on the data.

In Zenoh Flow, applications describe their graph in a declarative manner and each component is loaded dynamically, and the framework will create the link between the different components in order for the application to run seamlessly across the infrastructure. Zenoh Flow can be used to run multiple NI algorithms in parallel with the same input data, e.g., so as to compare accuracy, or to achieve online training while using an "old" model.

Thus, looking at the group of NI algorithms proposed in the above, energy-aware orchestration (Section 3), network state and capacity forecasting (Section 4), automated anomaly detection (Section 5), and self-learning MANO (Section 6), it is clear that an NI solution can be decomposed into graph of components, in which NI algorithms act as operators to compute data. Following are some of the benefits that using a dataflow programing approach can bring to the NI algorithms' implementations:

- Cloud-to-Thing compatible: having a framework that is designed to facilitate communication along the Cloud-to-Thing continuum is critical, and Zenoh Flow is also able to seamlessly run a dataflow graph where the computational nodes are running on different machines. Similarly, migrating a computational node, performing load-balancing or adding redundancy, become transparent procedures for an application developer.
- Feature-rich: Zenoh Flow offers a rich set of features to facilitate the NI algorithms creation and management.
  - automatic timestamps and end-to-end deadlines: to keep track when the data arrives, and to send notifications in case a deadline is missed;
  - input rules, also referred to as firing rules: used to specify under which conditions a computation can be triggered and how the inputs will be processed;
  - loops: can be used to support RL based algorithms;
  - logging and replying: used to log an execution and replay it later autonomously.
- Reusable: by abstracting communications we have the possibility to develop operators that are independent of any underlying infrastructure or dataflow graph. As a result, operators can be composed and reused in any Zenoh Flow dataflow graph.
- Declarative: a dataflow graph in Zenoh Flow is explicitly declared in a Yet Another Markup Language (YAML) file, as opposed to being implicit (such as with Robot Operating System (ROS) or ROS2) or specified directly in a program.

[1] https://github.com/eclipse-zenoh/zenoh-flow

3 Energy-aware VNF orchestration

B5G brings advanced capabilities to many application domains. Managing such demanding services is foreseen to be realized in a granular manner through the emerging architectures of microservices within a NF virtualization infrastructure (NFVI); this allows for several benefits but also raises novel challenges. The goal here is to focus on the energy-aware orchestration of services at different levels of a virtualized next-generation mobile network.

In this section, we present the work that has been carried out related to 1) energy aware VNF orchestration as part of studying the challenges of real-time control; and 2) VNF intelligence as a part of the design and development of NI-assisted functionality for B5G service and resource orchestration and management. Specifically, two different levels of virtualized services orchestration are presented: virtualizing NFs at user/data level (Section 3.1) and virtualizing the RAN (Sections 3.2 and 3.3).

3.1 Energy-aware VNF orchestration in heterogeneous network infrastructures

3.1.1 Problem description

As a result of the softwarization of B5G networks, enabled by Software Defined Networking (SDN) and NFV, it is now feasible to use general purpose resources (e.g., VMs) to implement the VNFs required by the different services. Without loss of generality, usually placement decisions focus only on computational and communication resources (e.g., VMs and the links connecting them). However, although VNFs are typically conditioned by a NFVI, other features should be included as part of the model used for their placement, such as the properties of each VNF, the KPI requirements of each service, the capabilities of VMs and physical devices (e.g., access points and datacenters) and their latency, just to mention a few. An optimal placement of VNF service functions in terms of energy consumption, helps to improve scalability of applications and contributes to reduce their response time and energy consumption. However, the high diversity of hardware and software technologies involved in VNFs impose some new challenges when we optimize their placement.

One way to address VNF placement as part of their orchestration in mobile networks is to reformulate, adapt and extend existing solutions applied in similar application domains, such as IoT and Cyber Physical Systems (CPSs) to the specific needs and requirements of VNFs. IoT and CBSs demand high performance computational resources capable of processing a large amount of data produced by a myriad of devices. To promote resource-efficient and sustainable solutions in terms of energy and latency costs, several approaches propose shifting the provision of services from the cloud to Internet edge. However, the heterogeneity of devices, the strict requirements of applications, and the sharing of software and hardware infrastructure can make this process difficult. In a previous work [4] we propose an offloading process that applies VMs to cope with the diversity of applications and infrastructures in the task offloading decision, considering the nodes' hardware and software characteristics (typically disregarded in code offloading approaches). Task offloading is supported by two modules, which firstly adapt the application task implementations to the infrastructure capabilities; and secondly assign application tasks to nodes, to minimize the energy consumption while assuring the QoS.

In summary, we propose energy aware VNF placements (as an initial step to VNF orchestration) by (i) modeling VNFs as a set of reusable service functions, including the definition of resource demands, energy cost and dependencies on the hardware and software of NFV and physical infrastructures; and (ii) by automating the optimization of VNF orchestration in a heterogeneous mobile infrastructure to reach a certain QoS (e.g., minimize power consumption) (see Figure **3.1.1**).

3.1.2 Modeling VNFs for energy-aware placement

Although the energy consumption of VNFs is typically affected by the hardware and software characteristics of the nodes in which they are executed, the heterogeneity of the infrastructure is often neglected in the energy-aware placement models. So, as a starting point we propose modelling the diversity of hardware and software energy-related characteristics of the infrastructure, to solve the problem of finding the underlying physical resources so as to perform an optimal energy-aware instantiation of VNFs service chains (see Figure **3.1.1**).

The use of multi-layer models [5] allows us to distinguish between the hardware and software characteristics to reuse and facilitate the configuration of a heterogeneous network infrastructure. The infrastructure models are extensible and reusable for any VNF.

In the model, network nodes are characterized by a set of hardware and software features: type of devices, computing capacity, amount of memory, sensing units, network capabilities, Operating System (OS), virtualization technologies supported, etc. These features are directly related to the type of service functions that can be placed on them and are intended to contain relevant information that will be used to predict the resource allocation, latency, and energy consumption of VNFs. For instance, the model

includes non-discrete numerical features of the nodes (e.g., CPU frequency), whose value is important to predict energy consumption and execution time of service functions.

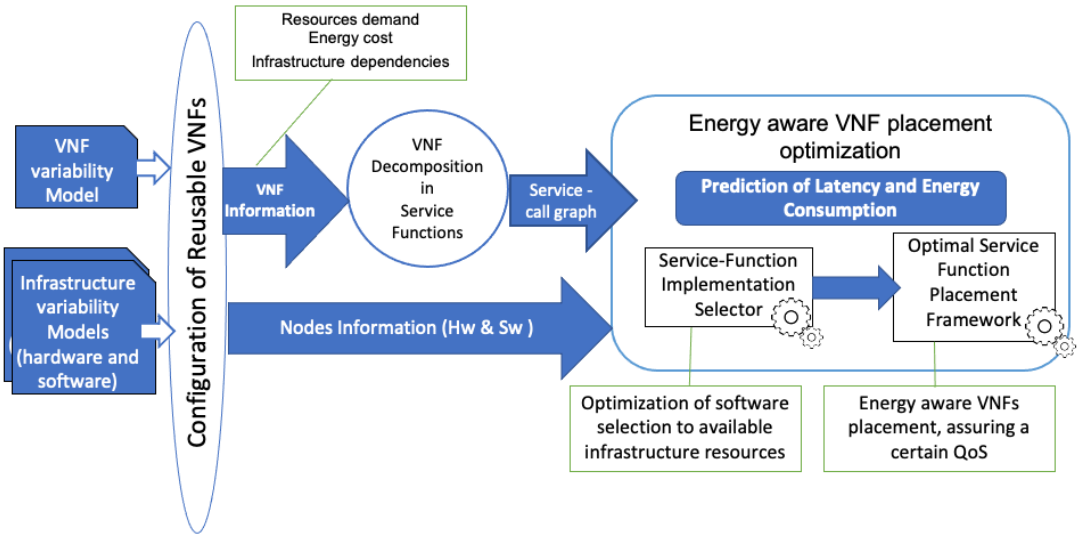


Figure 3.1.1. *The energy aware optimization of VNFs placement.*

IoT gateways (routers with processing capacity), cloudlets and vBSs located on the network are some examples of nodes that can be part of the infrastructure. The infrastructure software model includes information about the behavior of nodes, which can be defined by their role (such as *computing nodes* (capable to receive tasks) and *interactive nodes* (which are used directly by the users).

The Random Access Memory (RAM) and Hard Disk Drive (HDD) define the amount of memory and storage of nodes. The computation power is defined by their CPU, and optionally, GPU. Technically speaking, CPUs are defined by their number of cores and their frequency (in Hz) and GPUs by their RAM and frequency. Since devices may have several connectivity capabilities, this characteristic is modelled as a feature with cardinality. Each network connection is also characterized by their upload and download transmission rates R_{Tx} and R_{Rx} respectively, bits/sec) and their upload/download transmission powers P_{Tx} and P_{Rx} (Watts) (assumed like constants [6] [7] involved in the latency and energy consumption of communications). In case of multihop networks, $nHops$ represents the maximal and minimal number of hops between nodes. The location of the devices is determined by their latitude and longitude. Optionally, devices may have a set of sensing units associated: temperature, proximity sensors, microphones, or cameras are some examples. Other remarkable features included in the hardware model are the effective switched capacitance K, the importance of saving energy ew in the node and the type. The value of K depends on the chip architecture in the hardware and it directly influences the energy consumption of computing tasks [8] [9], while ew (with a value between 0 and 1) allows to manually set how important saving energy is in the node. Finally, the power supply determines if the device is plugged into the electrical power or is battery-operated.

Regarding the software characteristics of nodes, the model provides information about the OS and third-party libraries and enabling virtualization and container technology supported and the platform configuration (for cloud devices). Regarding virtualization, we distinguish between hardware/OS virtualization and lightweight virtualization based on containers. In the first case, software applications run on virtual hardware, allowing to define several machines inside the same device, which are considered as infrastructure's nodes (these are modelled using a link between these features and the hardware model referred above). These virtual nodes are constrained by the resources of the physical device where they are instantiated. This is managed by adding the identifier of the physical node to the hardware feature model with the physical information. This virtualization feature allows to select the technology used for virtualization with the aim of scaling up VMs on demand. Contained-based virtualization emulates an OS rather than the underlying hardware. As before, we define links between the models to create a relationship between features to configure the OS of containers, which avoid repeating parts of the models. It is possible to define runtime options related to memory, CPUs, and GPUs, limiting its usage. Once again, knowing their technology allows the instantiation of new containers on demand. Container technologies (e.g., Docker) allow managing the resources among containers instantiated in the same node according to the containers' workload. Nevertheless, they do not permit workload sharing among containers running on different nodes. This is the aim of orchestration systems for distributed architectures, which are modelled by adding the identity of the nodes subscribed to the orchestration. As for the network connectivity, nodes may have one or more Virtual Networks (VNs)

associated. The deployment infrastructure may include cloud nodes, that will be described by the platform and communication type to adapt the communication of the VNF service functions to them. This type of nodes can model both services in data centers in the core of the Internet, and small data centers in the edge of the Internet (cloudlets), differing in latency and computation characteristics.

3.1.3 Automation of energy aware VNF placement

In our approach, the automation of the assignment of tasks to devices (i.e., equivalent to VNF placement) is supported by two modules that implement two algorithms that assist developers in (a) the selection of the most appropriate set of tasks to carry out each application task according to the features of the existing infrastructure; and (b) the assignment of tasks and resources among the infrastructure nodes according to their current status and available resources (workload) (see Figure **3.1.1**). These modules use constraint programming, concretely a Satisfiability Modulo Theories (SMT) solver to provide an optimal solution to the task offloading problem. Our goal is to reformulate and adapt the algorithms of these modules to deal with VNF placement.

3.1.3.1 VNF decomposition in service functions

To deal with the relationships among VNF which identify the constraints imposed to the network infrastructure, we plan to use a service-call graph, which is typically a finite directed graph with no directed cycles. The vertices represent each VNF's interface, and the edges represent their dependencies.

This representation allows to detect VNF's service function whose beginning of execution depends on other VNFs (sequential dependency), and the existence of parallel service functions. Typically, VNFs have functionalities with time restrictions. Corresponding interfaces are grouped into sets with sequential dependency. At this point, the VNF's timeline should be determined, including the maximum time to be completed. These restrictions are specific to the VNF and must be fulfilled for all service tenants.

The models proposed are used to determine the latency of computation and data transmission associated with the VNFs in the nodes of the edge. The characteristics of the nodes along with the ones of the VNFs will be used to predict the latency associated for computation and data transmission. As an example, the equations used to calculate the task execution time used in a proposal for the deployment of application tasks are shown below. The first line of the following equation shows the expression used to calculate the computation time $tCompu_{t,i}$ (sec) of a VNF i by node n given by the relationship between the estimated number w_i of CPU cycles associated to the VNF i and the CPU power F_n (cycles per second) of node n. The communication time $tCommu_{i,j,n,z,ct}$ (sec) of the output data of VNF i sent to VNF j is given by the sum of the relationship between the amount of bits to send $c_{i,j}$ and the minimum between the upload transmission rate R_n^{Tx} of the sender node n) and the download transmission rate (R_z^{Rx} of the receiver node z) for a given connection type ct - e.g., WiFi 2.4 GHz, plus the propagation delay $t_{n,z}^{prop}$ (sec) between n and z - assumed to be 0 between edge devices [6] [7] - as seen in the second expression of following equation:

$$tCompu_{i,n} = x_{i,n}\frac{w_i}{F_n}$$

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

where $x_{i,n}$ is 1 if VNF i is assigned to node n, and 0 otherwise; $h_{i,j}$ is 1 if node n and z are not the same.

The propagation delay is set as the half of the mean Round Trip Time (RTT) obtained by pinging from n to z and considered constant [10]. Note that the first part of the equation, intended to determine the computation time, includes the frequency at which the node works. In case of using solutions to limit the node's resources (e.g., VMs, containers), the frequency to be used in the expression will be the one associated with that VM/container.

The above equation would be used to determine whether the deployment of a VNF at a given node would meet the desired QoS. In this sense, the execution of a given function would be associated with a maximum execution time (considering computation and data sending), i.e., the sum of the result of both expressions must be less than the maximum time. The proposed mechanisms must guarantee compliance with this QoS.

Regarding energy consumption, the first line of the following equation shows the expression used to determine the energy consumption of computation J required by node n to compute the VNF i $eCompu_{i,n}$. The energy consumed by task i to communicate with task j (using connection type ct) is presented in second and third expressions of the equation, that calculate the energy consumption for data sending and receiving in nodes n and z (sender and receiver nodes, respectively) [8]:

$$eCompu_{i,n} = x_{i,n}\kappa_n w_i F_n^2 ew_n$$

$$eSend_{c_{i,j},n,ct} = x_{i,n} h_{i,j} P^{Tx}_{n,ct} \frac{c_{i,j}}{R^{Tx}_{n,ct}} ew_n$$

$$eRecp_{c_{i,j},z,ct} = x_{j,z} h_{i,j} P^{Rx}_{z,ct} \frac{c_{i,j}}{R^{Rx}_{z,ct}} ew_z$$

where κ is a constant that depends on the hardware and directly influences the energy consumption of computation [8].

Although the energy consumption expression will be used to minimize the energy cost of the whole infrastructure, usually saving energy in some nodes (i.e., in battery powered nodes) is more important than for others. Other times, it is preferable to run the maximum number of functions on certain nodes (i.e., nodes powered by solar panels), as they minimize the carbon footprint. This is the aim of the *ew* (energy weight) variable, whose value is any number between 0 and 1. The value of this parameter increases accordingly with the importance of minimizing energy consumption at the node.

While the energy consumption in the nodes is influenced by several factors,, e.g., allocation of storage and RAM, the CPU usage is the most influential factor [8]. Note that, once again, the frequency at which the node works is included in the expression related with the computation $eCompu$. As in the execution time expression, in case of using solutions to limit the node's resources (e.g., VMs, containers), the frequency to be used in the expression will be the one associated with that VM or container. The aim is providing an accurate energy consumption in case that the frequency of the node is limited, since frequency is one of the most influential values in the energy consumption.

3.1.3.2 Energy-aware optimal placement algorithms

At this point, the service function's interfaces, and their timeline are determined. Prior to looking for an optimal placement, we plan to consider alternative service implementations differing in the energy consumption demands and the QoS offered.

Then, we plan to adapt the Tasks Implementation Selector (TIS) module, which evaluates alternative implementations of application tasks in terms of energy consumption, latency, and resource allocation, to VNF requirements and resources. Managing different implementations of the same VNF task allows: (i) to execute VNFs regardless of the mobile network infrastructure, as it enables adapting the software product's components to the available infrastructure; (ii) to use implementations optimized to the resources of the target node (e.g., lighter versions of software for battery-powered devices); and, in case of infrastructures composed of several nodes, (iii) to take advantage of the decomposition by selecting the most appropriate node to place each VNF's service function implementation.

In our previous work we addressed the implementation decision problem as a constraint-satisfaction problem that is resolved with a SMT-based approach. SMT is a formalized approach to constraint programming. Formalized as a form of the constraint satisfaction problem, (i) the algorithm of the solver always returns a solution, which guarantees that the assignment is feasible or the impossibility to place the VNF if no solution is found; and (ii) a large number of constraints (required to solve the problem at hand) help SMT-solvers to reduce the search space and to find the optimal solution faster [11] [12] However, *the flexibility of our approach allows to use other mathematical models for optimization for VNF placement*.

In the current version of the module the characteristics of the nodes included in the models, along with the ones of the tasks, are used to predict their latency and the energy consumption associated. The computation time $Tcomp_{i,n}$ (seconds) and the communication time $Tcomm_{i,j,n,z}$ of a task i by node n are calculated using the expressions given above. The energy consumption in the nodes is influenced by several factors, such as the usage of CPU, storage, and RAM, being the CPU usage the most influential. The first expression of the following equation shows the expression used in this work to predict the computation energy consumption (J) required by node *n* to compute task *i*. The energy consumption derived from communication between tasks in different nodes is given by the sum of the energy consumption in the sender and receiver nodes (second expression of Equation:

$$Ecomp_{i,n} = x_{i,n} \kappa_n w_i F_n^2 ew_n$$

$$Ecomm_{i,j,n,z} = x_{i,n} h_{i,j} P^{Tx}_n \frac{c_{i,j}}{Min(R^{Tx}_n, R^{Rx}_z)} ew_n + x_{j,z} h_{i,j} P^{Rx}_z \frac{c_{i,j}}{Min(R^{Tx}_n, R^{Rx}_z)} ew_z$$

The pseudo-code of the algorithm used by the TIS module is shown in the following figure. The algorithm receives as input the information of the nodes, the service-call graph of interfaces (with the time restrictions) and the number of tenants to support. As a result, it returns the configuration of service functions with the minimal energy consumption.

Module 1 Tasks Implementation Selector (TIS)

```
Data: Nodes; possibleConfigurations; timeRestrictions; nUsersToSupport
Result: Min(configurationsConsumption)
foreach configuration ∈ possibleConfigurations do
    Tasks ← configuration;
    opt = Optimize();
    opt.add( foreach task ∈ Tasks{
        assignedNode(task) ≠ null } );
    opt.add( foreach task ∈ Tasks {
        Sum(assignedRAM(task,assignedNode(task))<node_RAM*nUsersToSupport}});
    opt.add( foreach task ∈ Tasks {
        assignedRAM(task, assignedNode(task)) ≥ task_RAM });
    opt.add( foreach task_i ∈ Tasks {
        foreach task_j ∈ Tasks {
         if (hasToSendData(task_i,task_j) & assignedNode(task_i) ≠
            assignedNode(task_j)) then {
          connectionUsed(task_i) ≠ null }}} );
    opt.add( foreach task_i ∈ Tasks {
        foreach task_j ∈ Tasks {
         if (hasToSendData(task_i,task_j) & assignedNode(task_i) ≠
            assignedNode(task_j)) then {
          connectionUsed(task_i) ∈ assignedNode(task_i).connections ∩
            assignedNode(task_j).connections} else {
          connectionUsed(task) = Internet }}} );
    foreach tr ∈ timeRestrictions do
        opt.add(Sum[ foreach task_i ∈ tr {
            foreach task_j ∈ tr {
             timeComputation(task_i, assignedNode(task_i)) + tCommunication(
              task_i, task_j, assignedNode(task_i), assignedNode( task_j),
              connectionUsed(task_i) }}] ≤ tr.time);
    end
    opt.add( foreach task ∈ Tasks {
        task_requirements ⊆ assignedNode(task)_features } );
    opt.add(communicationEnergyCost = Sum [
        foreach task_j ∈ Tasks {
         foreach task_i ∈ Tasks {
          EnergyCommunication( task_i,task_j,assignedNode(task_i),
          assignedNode(task_j), connectionUsed( task_i))) }}]);
    opt.add(computationEnergyCost = Sum [
        foreach task ∈ Tasks {
         energyComputation(task, assignedNode(task)) }]);
    opt.minimize(communicationEnergyCost + computationEnergyCost);
    sat = opt.check(); // Checking the satisfacibility
    if sat then
        result = opt.solve();
        configurationsConsumption.add(configuration.id,
            result.communicationEnergyCost + result.computationEnergyCost);
    end
end
return(Min(configurationsConsumption));
```

3.1.3.3 *Energy aware VNFs placement*

Once selected the implementations with the lowest energy consumption, the module Optimal Service Function Placement Framework (OSFPF) will take as input the output of the TIS to select the node that best fit the instantiation of each service function. This placement decision considers the current available capabilities (computation, memory, storage) of each node of mobile infrastructure and the energy consumption information. The algorithm of this module will be based on the reformulation of the algorithm of the Optimal Task Assignment Framework (OTAF) module.

Module 2 Optimal Task Assignment Framework (OTAF)

```
Data: Nodes; Tasks; timeRestrictions; nUsersToSupport
Result: assignedNode(Tasks); RAMassigned(Tasks, Nodes); connectionUsed(Tasks)
Nodes ← currentStatus;
// Parameters to optimize:
assignedNode(task);// Optimal node to execute task
assignedRAM(task, node);// RAM allocated in node to task
connectionUsed(task)// connection used to send the data of task
opt = Optimize();
opt.add( foreach task ∈ Tasks{
    assignedNode(task) ≠ null } );
opt.add( foreach task ∈ Tasks {
    Sum(assignedRAM(task,assignedNode(task))<node_RAM*nUsersToSupport}});
opt.add( foreach task ∈ Tasks {
    assignedRAM(task, assignedNode(task)) ≥ task_RAM });
opt.add( foreach task_i ∈ Tasks {
    foreach task_j ∈ Tasks {
     if (hasToSendData(task_i,task_j) & assignedNode(task_i) ≠
        assignedNode(task_j)) then {
      connectionUsed(task_i) ≠ null }}} );
opt.add( foreach task_i ∈ Tasks {
    foreach task_j ∈ Tasks {
     if (hasToSendData(task_i,task_j) & assignedNode(task_i) ≠
        assignedNode(task_j)) then {
      connectionUsed(task_i) ∈ assignedNode(task_i).connections ∩
        assignedNode(task_j).connections} else {
      connectionUsed(task) = Internet }}} );
foreach tr ∈ timeRestrictions do
    opt.add(Sum[ foreach task_i ∈ tr {
        foreach task_j ∈ tr {
         timeComputation(task_i, assignedNode(task_i)) + tCommunication(
          task_i, task_j, assignedNode(task_i), assignedNode( task_j),
          connectionUsed(task_i) }}] ≤ tr.time);
end
opt.add( foreach task ∈ Tasks {
    task_requirements ⊆ assignedNode(task)_features } );
opt.add(communicationEnergyCost = Sum [
    foreach task_j ∈ Tasks {
     foreach task_i ∈ Tasks {
      EnergyCommunication( task_i,task_j,assignedNode(task_i),
      assignedNode(task_j), connectionUsed( task_i))) }}]);
opt.add(computationEnergyCost = Sum [
    foreach task ∈ Tasks {
     energyComputation(task, assignedNode(task)) }]);
opt.minimize(communicationEnergyCost + computationEnergyCost);
sat = opt.check(); // Checking the satisfacibility
if sat then
    solution = opt.solve();
end
return(solution);
```

3.1.3.4 *Proof of concept and Evaluation*

We have applied our approach to a real case of an academic campus where several devices, those typical of 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. Experimentation studied the reduction of power consumption obtained by application task offloading and measuring the execution time to find a task offloading solution for different problem sizes, and considering this infrastructure is shared by several applications at the same time.

The IoT gateways periodically send the information collected from the sensors to a cloudlet, which updates its databases with up-to-date measures. The infrastructure also contains 3 computers acting as edge nodes, and a cloud server. For our experiments, the characteristics of the 12 nodes (the user node, 10 edge nodes and 1 cloud node) have been randomly generated. Concretely, the CPU speed of IoT gateways ranges from 1 to 1.6 GHz, the cloud server from 2.4 to 4 GHz, the cloudlet from 2 to 2.4 GHz, and the rest of edge devices from 1.6 to 2 GHz.

R^{Tx} and R^{Rx} have been set between 100 and 150 Mbps for edge devices and from 8 to 10 Mbps for cloud devices. The propagation delay between the cloud device and the rest of nodes has been set from 0.02 to 0.1 s. κ has a value between $1{\cdot}10^{-9}$ and $1{\cdot}10^{-11}$ [9] [7]. P^{Tx} and P^{Rx} have values between 1 and 1.5 W in all cases [7]. Finally, ew has been set to 1 for all nodes. To avoid the mobility problem in edge computing [13], the task assignment relies on Wireless Local Area Network (WLAN) and the Internet for data transmission.

Experiments consider two scenarios. In the first one (Scenario 1), the nodes reserve a fixed number of resources (2 GB of RAM and one CPU core) dedicated to computation of offloaded tasks. In the second one (Scenario 2), the nodes do not reserve specific resources for offloaded tasks, so the feasibility of the task offloading process and the reduction in energy consumption will depend on the current nodes' workload (randomized in each experiment from 0 to their maximal capabilities). Each test is performed 30 times. Table 2 shows the Reduction of Energy Consumption (REC) obtained in the experiments performed for both scenarios. For Scenario 1, three rows detail the task offloading solutions (distribution of tasks in the different nodes, in columns N1 to N12, and RAM used) and the REC for three different states of workload in the infrastructure, which depends on the number of users: $user \leq 13$; $14 \leq users \leq 26$ and $27 \leq users \leq 39$.

These results allow to compare the energy consumption of distributing the application (VNFs) according to the assignment solution obtained by our approach, with the energy consumption of running the entire application on the user device. This REC is given as a percentage (%). The reduction is calculated considering the energy consumption of all nodes of the infrastructure (including the user one) and considering just the energy consumed in the user node columns named REC (all nodes) and REC (user node) respectively.

Table 2. *Result of reduction in the energy consumption (REC) in simulated scenarios of AR.*

SCENARIO	PARAMETERS	REC [NOTE] (ALL NODES)	REC (USER NODE)
SCENARIO 1: FIXED RESOURCES ALLOCATION	users ≤ 13 (state 1)	**48.0%**	**62.2%**
	14 ≤ users ≤ 13 (state 2)	**43.5%**	**62.2%**
	27 ≤ users ≤ 13 (state 3)	**37.3%**	**62.2%**
SCENARIO 2: NON-FIXED RESOURCES ALLOCATION	REC considering all nodes (Avg/Max/Min/Std)	**41.1 / 55.2 / 33.6/ 9.3 %**	
	REC in the user node (Avg/Max/Min/Std)	**56.5 / 65.1 / 41.2 / 8.8 %**	
[NOTE] REC: REDUCTION IN THE ENERGY CONSUMPTION			

3.2 Energy-driven vRAN orchestration

Virtualization is considered today the most promising approach for bringing cellular networks up to speed with the demanding services they aspire to support. The latest frontier in this endeavor is virtualizing the RAN by turning the BSs into fully-softwarized stacks that can be deployed in diverse platforms such as commodity servers, small embedded devices, or even moving nodes (*cells-on-wheels*). This paradigm shift is expected to offer the much-needed performance flexibility, facilitate the ongoing network densification, and reduce capital and operating expenses. Hence, not surprisingly, it has spurred numerous industry efforts to build BS stacks, fully-open RANs, and even launch extensive field tests.

However, designing and operating vRANs is far from trivial, since the vBSs differ significantly from their hardware-based counterparts. On the one hand, vBSs are more controllable as one can tune their parameters (transmission power, modulation schemes, etc.) in real time based on the network needs. On the other hand, their softwarization and diverse platforms render less predictable their performance and power consumption. The latter is particularly important both for economic reasons but also because the vBSs, most often, operate under tight energy budgets. Hence, traditional radio control policies run the risk of under-utilizing this new type of BSs, or rendering vRANs economically unsustainable. Clearly, in order to unleash the full potential of vRANs we need to answer two key questions: *(i) what is the performance and energy consumption profile of vBSs?* and *(ii) how can we optimize their operation using an adaptive and platform-oblivious approach?*

We first analyze experimentally the vBS operation using different platforms and scenarios. Our results shed light on the relationship between performance (throughput), power consumption, and vBS controls such as the Modulation and Coding Schemes (MCSs) and spectrum allocation. For instance, we find that the BaseBand Unit (BBU) consumes power comparable to wireless transmissions, and the vBS power and

throughput are affected by the configurations in a non-linear/non-monotonic fashion. These results depend also on the hosting platform. Moreover, we observe that UpLink (UL)-related computations consume more power and are more sensitive on MCS and SNR variations, than the respective DownLink (DL)-related computations; a finding attributed to the heavier UL decoding. Our analysis is centered on energy, which is the bottleneck vBS resource that affects both its computations and transmissions (see Figure **3.2.1**).

3.2.1 Experimental Analysis

We performed exhaustive experiments using an srsRAN-based[2] vBS testbed. Our testbed is shown in Figure **3.2.1** and comprises a vBS, the User Equipment (UE)[3], and a digital power meter. Both the vBS and the UE consist of an Ettus Research USRP B210 as RU, srseNB/srsUE (from srsRAN suite [14]) as BBU for both the eNB and UE, and two small factor general-purpose PCs (Intel NUCs with Central Processing Unit (CPU) i7-8559U@2.70GHz) deploying each respective BBU and the Near Real Time (NRT) RAN Intelligent Controller (RIC) of O-RAN [15]. The vBS and the UE are connected using SMA cables with 20dB attenuators, and we adjust the gain of the RU's Radio Frequency (RF) chains to attain different Signal to Noise Ratio (SNR) values. Without loss of generality, we select a 10-MHz band that renders a maximum capacity of roughly 32 and 23 Mbps in DL and UL, respectively. We use the power meter GW-Instek GPM-8213 to measure the power consumption of the BBU and the RU by plugging their power supply cable to a GW-Instek Measuring adapter GPM-001. Finally, we have integrated O-RAN E2's interface and the ability to enforce control policies *on-the-fly* in srseNB.

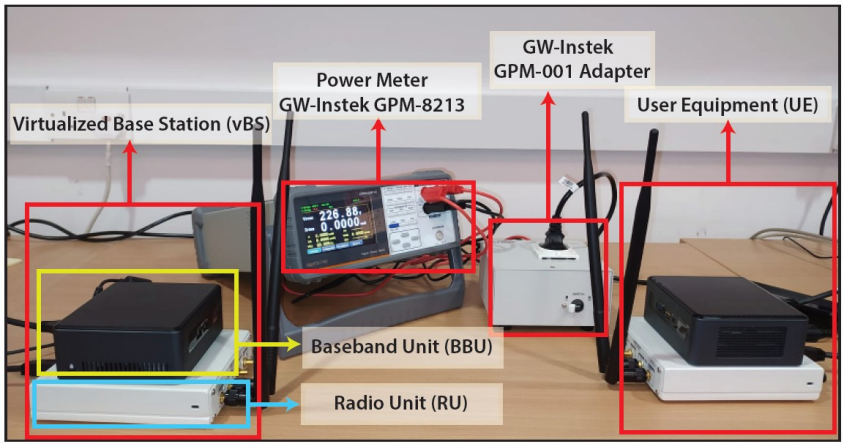


Figure 3.2.1. *Toy experimental setup.*

3.2.1.1 BBU/CPU Power Cost & Impact of Platform

The first important finding is that the power consumption associated with BBU processing is *comparable* to the RF chain's transmission power. This result is consistent with previous studies; e.g., [16] estimated that 40% of a femtocell's power consumption is due to BBU. Figure **3.2.2** dissects the power consumption of a vBS deployed over a Small Factor (SF) PC into the share responsible by (i) the BBU´s CPUs[4], (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 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 **3.2.3** shows the BBU consumption over the baseline for different platforms.[5] 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.

[2] https://www.srsran.com/

[3] We use one UE emulating the load of multiple users.

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

[5] 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.

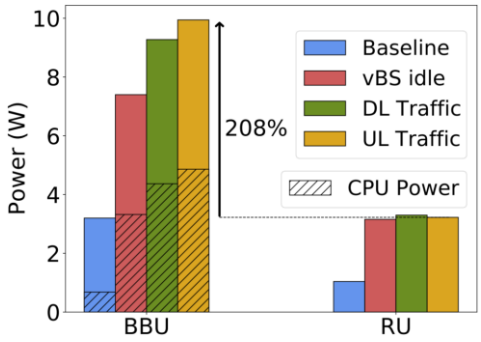


Figure 3.2.2. *Comparison of power consumption at: the BBU (Intel NUC i7-8559U@2.70GHz), the BBU's CPU, and the RU (an USRP SDR), with 20Mbps DL and UL traffic.*

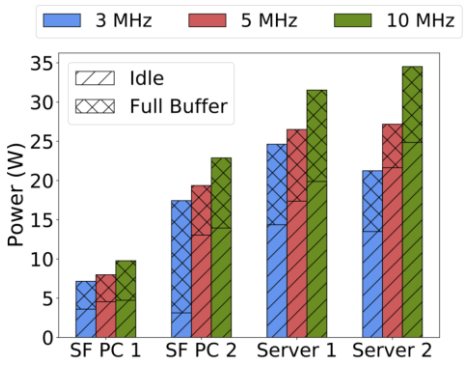


Figure 3.2.3. *Consumed power over the baseline for different radio bandwidths and hardware platforms. SF PC 1: Intel NUC i7-8559U@2.70GHz; SF PC 2: Intel NUC i7-8650U@1.90GHz; Server 1: Dell XPS 8900 i7-6700@3.40GHz; Server 2: Dell Aurora R5 i7-9700@3.00GHz.*

3.2.1.2 *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 **3.2.4**; and this induces a commensurate increase in power consumption, see Figure **3.2.5**. Besides, Figure **3.2.5** 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 [17] [18]. Hence, maximizing throughput does not only have an unpredictable effect on power, but it is indeed highly non-trivial.

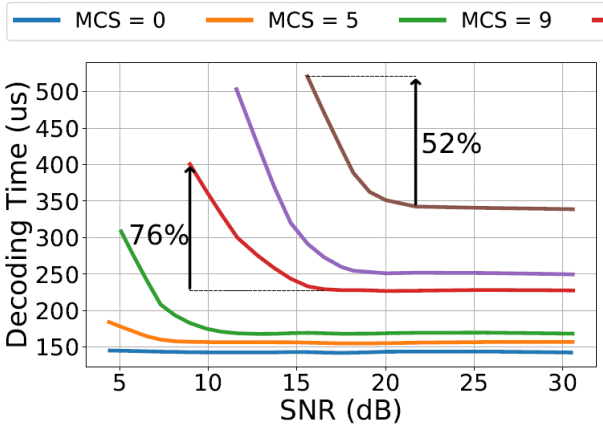


Figure 3.2.4. *vBS over SF PC 1 at full UL buffer. UL decoding time as a function of SNR and different MCS values.*

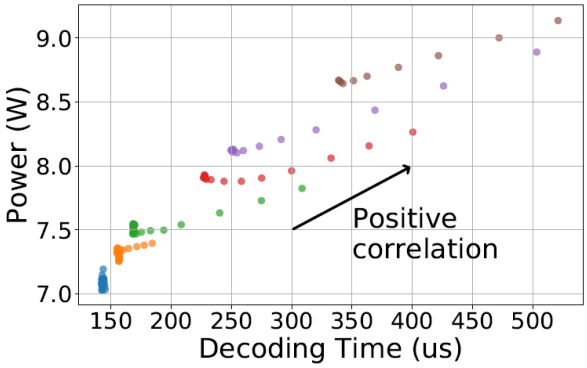


Figure 3.2.5. *vBS over SF PC 1 at full UL buffer. Power consumption as a function of the decoder performance (high correlation).*

3.2.1.3 *Configuration Options & Impact of Scheduler*

The above vBS control challenges are exacerbated by the plenitude of configuration options. Figure **3.2.6**, 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 **3.2.5**). 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 **3.2.7** 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*.

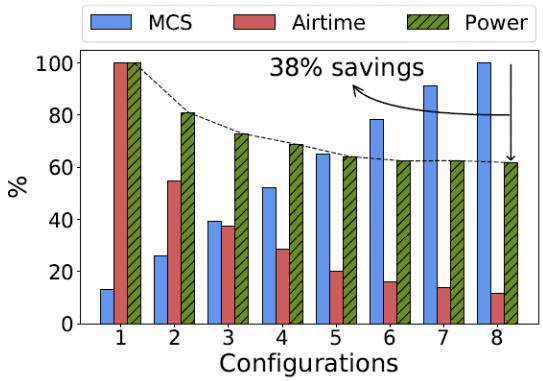


Figure 3.2.6. *8x combinations of normalized MCS and airtime providing 2.6Mbps in UL, and its associated power (idle mode power is subtracted).*

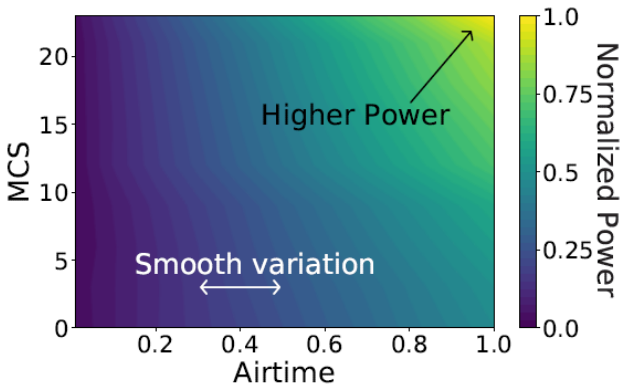


Figure 3.2.7. *Normalized power consumption at the BBU over baseline for full buffer UL transmissions and high SNR, as a function of MCS and airtime.*

3.2.1.4 Coupling of DL & UL Processing

Finally, Figure **3.2.8** shows the BBU power consumption when DL and UL traffic streams are processed separately and concurrently (UL+DL), for different MCSs and high SNR. We observe that the joint power is not the total sum of the separate components. For instance, for MCS 15, concurrent DL and UL processing consumes just 7.5% more than UL-only processing (and 26% over DL-only). This is because there are common power consumption factors in both links. This, in turn, makes it difficult to predict the overall vBS power consumption, given that the DL and UL can be configured separately. Also, note that UL power costs are higher and more volatile than DL, since decoding is more computationally demanding.

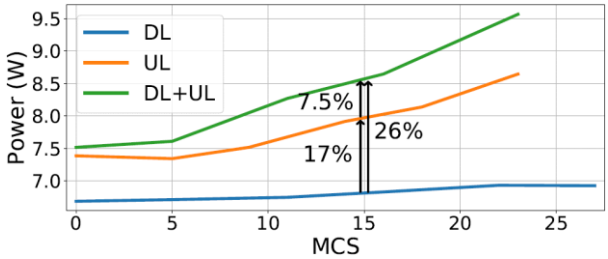


Figure 3.2.8. *MCS impact on BBU power consumption with high SNR.*

3.2.1.5 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. Hence, we propose the use of *online learning* to profile each vBS power cost and performance, and devise goal-driven configuration policies.

3.2.2 System Model and Problem Formulation

The experimental analysis provided above shed light on the relevant settings and performance metrics that should be considered to optimize networking performance and energy consumption. Motivated by these experiments, we next formalize mathematically an optimization problem. We consider a vBS comprising a BBU, which may correspond to a 4G evolved node B (eNB) or 5G next generation node B (gNB)[6] hosted in a cloud platform and attached to a RU, which are fed by a common and possibly constrained energy source. This type of BSs is relevant for low-cost small cells, Power-over-Ethernet (PoE) cells, and so on. Our goal is to use O-RAN's control architecture to select and adapt radio policies to system dynamics satisfying different energy-driven criteria. Figure **3.2.9** shows the high-level system architecture, which is O-RAN compliant [15]. The Learning Agent (LA) runs online algorithms within the NRT RIC in the system's orchestrator, and sequentially selects efficient *radio policies* every orchestration period t (in the order of seconds) *given the current context*. We hence formulate our problem as a contextual multi-armed bandit or *contextual bandit*.

[6] 5G decouples BBU in 2 logical functions, i.e., a Central Unit (CU) and a Distributed Unit (DU). Our scheme controls the DU, or both when co-located.

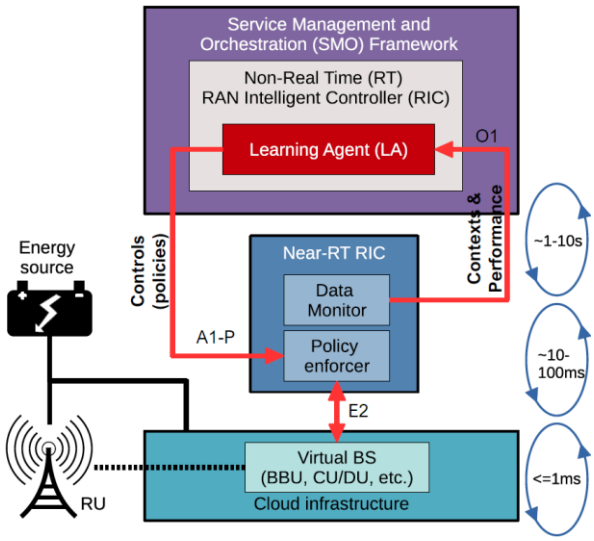


Figure 3.2.9. *O-RAN compliant system architecture and workflow.*

Contexts. We define the DL context at each period t as $\omega_t^{dl} \coloneqq \left[\overline{c}_t^{dl}, \widetilde{c}_t^{dl}, d_t^{dl}\right]$, where $\overline{c}_t^{dl}$ and $\widetilde{c_t^{dl}}$ are the mean and variance of the DL Channel Quality Indicator (CQI) across all users in the previous period; and d_t^{dl} is the *new* bit arrivals at the vBS DL aggregated across all users. Note that the DL CQI is sent periodically from the UEs to vBS through Uplink Control Information (UCI) carried by 4G/5G's Physical Uplink Shared Channel (PUSCH) or Physical Uplink Control Channel (PUCCH). Conversely, d_t^{dl} is measured by the vBS at the PDCP layer. Also, we define the UL context as $\omega_t^{ul} \coloneqq \left[\overline{c}_t^{ul}, \widetilde{c}_t^{dl}, d_t^{ul}\right]$. The UL CQI is measured by the vBS at MAC layer, and the new UL bit arrivals are estimated from the periodic Buffer Status Reports (BSRs) of the users (UEs). All these measurements are collected by the Near-RT RIC's Data Monitor (Figure 3.2.9) from the vBS using the E2 interface at sub-second granularity and are aggregated at the start of each orchestration period t. We denote the global context vector $\omega_t \coloneqq \left[\omega_t^{dl}, \omega_t^{ul}\right] \in \Omega$, where Ω is the context space.

Controls. We define the DL control $x_t^{dl} \coloneqq \left[p_t^{dl}, m_t^{dl}, a_t^{dl}\right]$ at period t, where $p_t^{dl} \in \mathcal{P}^{dl}$ is a *Transmission Power Control (TPC) policy* for the maximum allowed vBS transmission power, $m_t^{dl} \in \mathcal{M}^{dl}$ is the highest MCS eligible by the vBS (*DL MCS policy*), and $a_t^{dl} \in \mathcal{A}^{dl}$ is the maximum vBS transmission airtime (*DL airtime policy*). We define the UL control $x_t^{ul} \coloneqq \left[m_t^{ul}, a_t^{ul}\right]$, where $m_t^{ul} \in \mathcal{M}^{ul}$ and $a_t^{ul} \in \mathcal{A}^{ul}$ are the UL MCS and airtime policies.[7] We hence formalize each control at decision period t as a *radio policy* $x_t \coloneqq \left[x_t^{dl}, x_t^{ul}\right] \in \mathcal{X}$, where $\mathcal{X} = \mathcal{P}^{dl} \times \mathcal{M}^{dl} \times \mathcal{A}^{dl} \times \mathcal{M}^{ul} \times \mathcal{A}^{ul}$ is the control space. Once computed, the LA sends each radio control policy to the NRT RIC via O-RAN's A1-P interface, which is then applied to vBS. The UL policies are applied by configuring each UL scheduling at the vBS MAC layer.

Rewards. We denote $R^{dl}\left(\omega_t^{dl}, x_t^{dl}\right)$ and $R^{ul}\left(\omega_t^{ul}, x_t^{ul}\right)$ the DL and UL data transmission rates, and define the *reward* function $r(\omega_t, x_t)$ as:

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

where the logarithms are used to achieve fairness between the DL and UL flows – and to that end, one could use any other α-fair function. The reward is computed at the end of each period by the NRT RIC's data monitor and sent to LA. It is important to stress that in practice we can only hope to observe *noisy values* of these functions, even when their arguments are fixed, because naturally the system operation is stochastic and also the power measurements are noisy – as we have indeed seen in our experiments. Fortunately, our optimization framework can handle such impairments. Henceforth, we denote $R_t^{dl}\left(\omega_t^{dl}, x_t^{dl}\right)$, $R_t^{ul}\left(\omega_t^{ul}, x_t^{ul}\right)$ and $r_t(\omega_t, x_t)$ the sample at period t of the functions.

We formulate two energy-aware vBS control problems: (i) BP-vRAN (Balanced Performance – vRAN), which finds the Pareto-optimal trade-off between performance and power consumption; and (ii) SBP-vRAN (Safe Balanced Performance – vRAN), which maximizes performance subject to *hard* constraints on the power source. The former allows operators to balance performance and power expenses, while the latter is crucial for vBS running on power-constrained platforms, e.g., PoE cells.

[7] We do not define an UL TPC policy for simplicity because the users' transmission power has less impact on the vBS power than the MCS and UL airtime; but our framework can be readily extended to that.

3.2.2.1 Case 1: Balancing performance and cost (BP-vRAN)

We consider first the case where the power supply is scarce, or the operator simply wants to reduce the power costs. This can be achieved with a scalarized *objective function*:

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

where $P(\omega_t, x_t)$ is the vBS power consumption associated with the pair context-control (ω_t, x_t), $B(\cdot)$ corresponds to a smooth function that models the monetary cost associated with power consumption, and parameter δ prioritizes one over the other criterion based on the operator's preferences. Considering the noisy samples at period t $P_t(\omega_t, x_t)$ and $r_t(\omega_t, x_t)$, we define $u_t(\omega_t, x_t)$ as the objective function observation. In order to penalize power-consuming policies, we use a sigmoid function with sharpness and tipping parameters a and b:

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

When $a \to 0$, $B(\cdot)$ approximates a linear function, and the step function when a grows.

Following the standard approach in Bayesian bandit optimization, we use the cumulative contextual regret to assess the performance of our algorithm. We define the average T-period contextual regret:

$$R_T \coloneqq \sum_{t=1}^{T}\left(\max_{x' \in \mathcal{X}} u(\omega_t, x') \; - u(\omega_t, x_t)\right),$$

where $\max_{x' \in \mathcal{X}} u(\omega_t, x')$ yields the best decision for the current period, which we cannot calculate in practice since the objective function is unknown. Our goal, therefore, is to find a sequence of decisions $\langle x_t \rangle_{t=1}^{T}$ from set $\mathcal{X}$ which ensure asymptotically sublinear average regret, i.e., $\frac{\lim_{T\to\infty} R_T}{T} = 0$.

3.2.2.2 Case 2: Maximize performance subject to hard power constraints (SBP-vRAN)

When the vBS operates under a hard power budget $P_{\max}$, e.g., when PoE, the LA has to find the maximum-throughput configuration that respects this budget. Importantly, the LA needs to achieve that with a *safe* exploration of the configuration space $\mathcal{X}$ in order to satisfy the $P_{\max}$ threshold at any period, i.e., not only at the final optimal-operation stage. We define the respective regret:

$$R_T^S \coloneqq \sum_{t=1}^{T}\left(\max_{x' \in S_t(\omega_t)} r(\omega_t, x') - r(\omega_t, x_t)\right),$$

where in this case the decisions are selected from set

$$S_t(\omega_t) = \{\, x \in \mathcal{X} \mid P(\omega_t, x) \leq P_{\max} \,\}.$$

Our goal is to find a sequence $\langle x_t \rangle_{t=1}^{T}$, $x_t \in S_t(\omega_t)$, such that $\frac{\lim_{T\to\infty} R_T^S}{T} = 0$. Note that sets $S_t(\omega_t), \forall \omega_t$, are unknown since $P(\omega, x)$ is unknown, and thus need to be learned from the measurements $P_t(\omega_t, x_t)$. And, similarly, we only have access to r_t and u_t, i.e., the t-period noisy measurements, instead of the actual functions r and u.

3.3 Energy-driven joint vRAN & edge service orchestration

There is growing consensus that the next generation of mobile networks need to support Artificial Intelligence (AI) services *at the edge*. This involves the collection, transfer and processing of data flows, with the aim to provide real-time inferences to end users ranging from handheld and small IoT devices to moving nodes (e.g., drones). Representative examples are Mobile Video Analytics (MVA), which are used in Augmented Reality/Virtual Reality (AR/VR) services, cognitive assistance apps, surveillance systems, and other similar Artificial Intelligence (AI) services. The core task of MVA services is that user devices send video frames to the network, which needs to process them and transmit back accurately detected depicted objects or extract other important information. These services are equally exciting for the users as they are challenging for the networks to implement.

Namely, in MVA services the network's role is not confined to transferring data from one point to another, nor it suffices to support in-network processing of the data streams. Instead, the network needs to directly optimize the AI service performance, which involves the criteria of accuracy (confident inferences), end-to-end latency (fast inferences), and task throughput (inferences/sec) in a resource-efficient fashion. This latter requirement is crucial since AI services create voluminous data flows, involve heavy computations, and consume large amounts of energy, which is one of the most prevalent cost factors in mobile networks. Clearly, in order to realize edge AI services, we need to devise a rigorous methodology for controlling jointly the communication and computing resources of the network. *Our goal is to design and evaluate experimentally an optimization framework that orchestrates the resources of BSs and edge MVA-based AI services.*

To shed light on this problem, we have built a fully-fledged prototype system with a software-defined BS (using srsRAN suite [19]) and a Graphical Processing Unit (GPU)-enabled edge server that offers a 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.

In order to overcome these challenges, we propose an optimization framework for orchestrating jointly these decisions that is oblivious to the underlying hardware and user data streams.

3.3.1 Experimental analysis

We have performed an exhaustive set of experiments using a testbed. In a nutshell, the testbed is comprised of a 3GPP R10-compliant Long Term Evolution (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[8]. 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 available online[9] 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.

3.3.1.1 Precision, delay, and image resolution

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 and 2, respectively.

Performance Indicator 1 (Service Delay). End-to-end delay that includes the image pre-processing at the user side, its transmission, the processing at the server (GPU delay), and the return of the bounding boxes and labels.

Performance Indicator 2 (Mean Average Precision). The service's precision is estimated by the mean Average Precision (AP), a popular metric in computer vision for object recognition applications. On the one hand, precision is defined as the ratio of true positives over all positive classifications. On the other hand, the recall measures how well these positives are identified by calculating the ratio between true positives over the sum of true positives and false negatives. The Intersection over Union (IoU) measures the overlap between the calculated bounding box and the ground truth. IoU values above a threshold, which in our case is set to 0.5, trigger a true positive. Then, given a set of images, the average precision corresponds to the area below the precision-recall curve. Finally, the mean Average Precision (mAP) is calculated as the mean average precision over all object categories. In this way, the mAP spans between 0 (worst performance) and 1 (best performance).

According to our measurements, the most relevant feature that affects the mAP is the *image resolution*, defined in Policy 1.

Policy 1 (Image Resolution). Selected encoding of the image (number of pixels) generated by the user, which can be enforced by the service. In our experiments, the maximum (100%) resolution is 640x480 pixels.

We illustrate this in Figure ***3.3.1***, 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%.

[8] https://cocodataset.org/

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

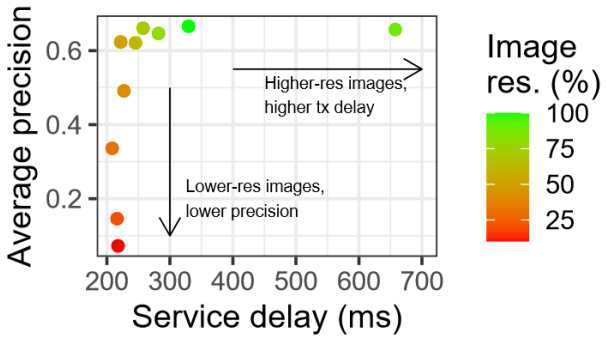


Figure 3.3.1. *Mean average precision (mAP) vs.service delay for images with different resolutions.*

3.3.1.2 Delay, energy consumption, and radio policies

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.

Policy 2 (Radio Airtime). This radio policy imposes a constraint on the amount of radio resources (duty cycle) the BS allocates to the service's traffic. This in line with the concept of network slicing in 5G. Due to the nature of this AI service, we focus on uplink communication.

Performance Indicator 3 (Server Power Consumption). The power consumption associated with the computational load of the service requests, mostly dominated by the GPU power consumption.

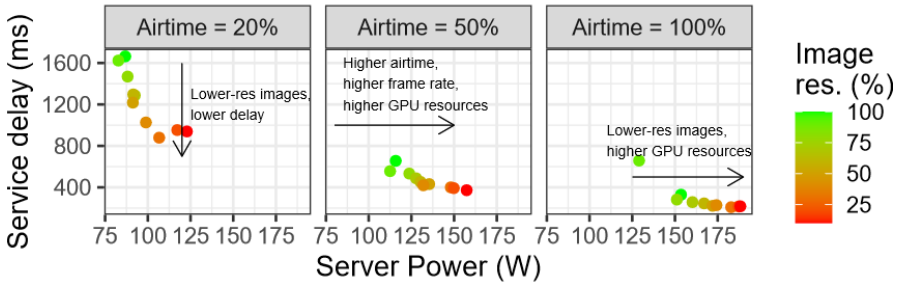


Figure 3.3.2. *Service delay vs. server's power consumption for images with different resolutions and radio policies.*

Figure **3.3.2** 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 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).

3.3.1.3 Delay, energy costs, and service policies

We study the impact of the computing allocation policies on the service quality of service. To this end, we define an additional configuration policy.

Policy 3. The server's policy is a GPU power limit that adapt the GPU's processing speed to meet the set power constraint.

In our experimental setup, the GPU speed can be set through a configuration parameter available in NVIDIA GPU drivers. Figure **3.3.3** (top) depicts, similarly as above, the service delay and the server's power consumption for several image resolution configurations. We now fix the airtime to 100% and vary the policy allocating computing resources. A higher amount of computing resources increases the server's power consumption, as we are relaxing the power limit imposed to the GPU. Like before, we observe that lower resolution images contribute to increasing the server's power consumption as the rate of requests also grows. However, it is interesting to note that higher resolution images ease the work on the GPU, as evidenced by Figure **3.3.3** (bottom), which shows the delay associated with the GPU tasks only. All in all, despite this improvement in the GPU delay, the corresponding increase in transmission delay when using higher resolution images dominates. It is important to observe, that while this is true in our experimental testbed, it may well be different for diverse deployments (e.g., a more energy-efficient GPU, or a higher-bandwidth RAN). This motivates the need for *learning* algorithms that are able to adapt to the different specific deployments.

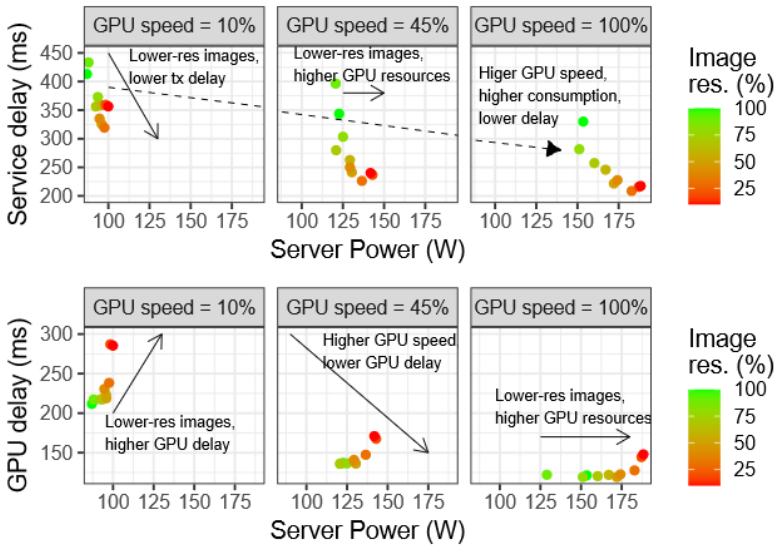


Figure 3.3.3. *Delay vs. server's power consumption for images with different resolutions and GPU policies.*

3.3.1.4 Precision, energy consumption, and image resolution

The above trade-off between service delay and the server power consumption (a cost to the service provider), certainly appears for other user-related metrics, such as the mAP introduced earlier. To assess this,

Figure ***3.3.4*** shows the mAP performance achieved by the service as a function of the server's power consumption for a variable set of image resolutions and the highest amount of radio and computing resources to minimize delay. The figure confirms that the service provider's cost is also dependent on the mAP performance. Importantly, however, the relationship with the mAP is substantially different to that with the service delay. In this case, higher mAP performance actually requires *less* amount of power consumption from the service provider. The reason lies upon the fact that higher resolution images (which render higher mAP) facilitate the object detection task, and hence, require less computing resources as shown in Figure **3.3.3** (bottom).

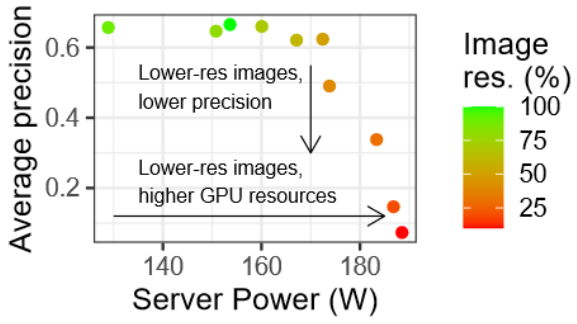


Figure 3.3.4. *Mean average precision vs. server's powerconsumption for images with different resolutions.*

3.3.1.5 *BS power consumption, radio policies, and image resolution*

Finally, the costs associated with the network operator (our third stakeholder) are necessarily driven by the amount of radio resources invested into the service's pipeline. To analyze this, we introduce an additional policy, motivated by [17] in the context of vRANs, which is defined as Policy 4, and Performance Indicator 4, reflecting part of operational costs of the network operator.

Policy 4 (Radio MCS). This policy imposes a constraint on the maximum MCS eligible by the BS to transport the service's data over the air.

Performance Indicator 4 (BS Power Consumption). Power consumption associated with processing the baseband unit in a vRAN environment.

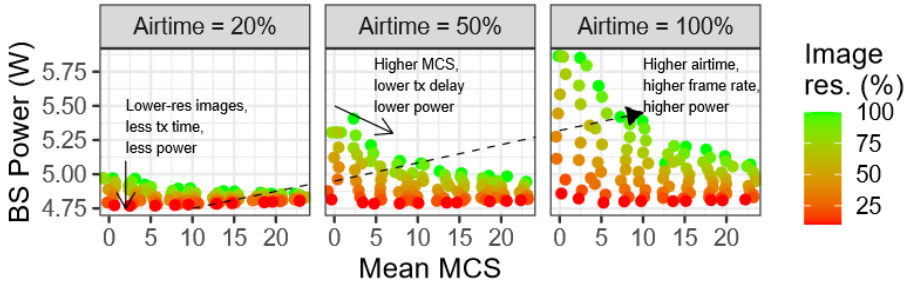


Figure 3.3.5. *BS power consumption vs. radio policies forimages with different resolutions.*

To analyze these, we plot in Figure **3.3.5** the power consumption measured at the baseband unit of the BS for a variable setting of airtime and MCS radio policies, and different image resolutions. We first observe that lower-resolution images consume less amount of radio resources and therefore have a smaller footprint over the BS's power consumption. Second, a higher investment on radio resources (airtime) actually induces higher power consumption because it allows the user to issue a higher rate of service requests (images). Finally, perhaps surprisingly, higher MCS policies actually cause *lower* BS power consumption. The reason is that the data load at the BS is relatively low compared to the amount of bandwidth available at the BS, e.g., higher resolution images with 100% airtime generate up to 2.8 Mb/s, compared to a capacity of around 50Mb/s (SISO LTE @ 20MHz bandwidth). In this scenario, despite the fact that LTE subframes modulated with higher MCS incur higher instantaneous power consumption, they process the load in shorter time, which pays off in terms of power consumption over the long run.

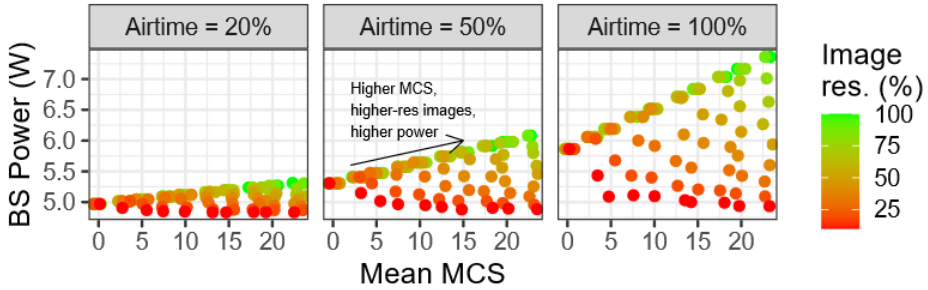


Figure 3.3.6. *BS power consumption vs. radio policies forimages with different resolutions and 10x higher load.*

From these results and the BS's point of view, there is no reason to use MCS lower than the maximum possible. However, this depends also on the traffic load, which may be very different for, e.g., multiple users or different services. To demonstrate this, we emulate a scenario with 10x more load, and present the same plot in Figure **3.3.6**. Differently to the previous case, we now observe that the MCS policy has a negative impact on the BS power consumption for higher-resolution images whereas lower resolution images cause lower power consumption for higher MCS policies. This motivates the need for learning algorithms that adapt to the service requirements and loads.

3.3.1.6 *Conclusions:*

Our system consists of a large number of highly coupled parameters with non-trivial relationship with the performance and cost. As a consequence, we resort to model-free RL methods, an area of machine learning applied in many control problems, to design a controller that adapts autonomously to context changes and the underlying platform.

3.3.2 System model and problem formulation

We consider a GPU-powered edge server providing an AI service through a RAN. Specifically, we consider an object recognition service that can be used, for instance, for security surveillance or fault detection in industrial chains. We assume a slice dedicated for this service is created including vBS and the edge server. The workflow is very simple: users capture images that are sent to the edge server through the UL of the radio interface of the vBS. Then, edge server's GPU processes the incoming data and generates a response, which is sent back to the users through the DL of the vBS.

We consider a LA that is in charge of controlling the GPU, the resources assigned to the network slice, and the amount of data sent by the user, via a set of *control policies*. We follow closely the framework of O-RAN [15]. That is, the LA runs in the Service Management and Orchestration (SMO) in the timescale of seconds. Because our LA provides both radio and edge computing control policies, it interacts with the NRT RIC and the edge orchestrator therein. Radio policies are in turn enforced by the NRT RIC by using O-RAN's E2 interface. Figure ***3.3.7*** summarizes the architecture of the system. Feedback from the data plane components are collected by the Edge orchestrator and NRT RIC (via O-RAN's O1 interface) and then fed to the LA. This allows us to use RL to solve our problem. Namely, we formulate the problem as a contextual multi-armed bandit or *contextual bandit*. We next formally define the contexts, control policies, and performance indicators.

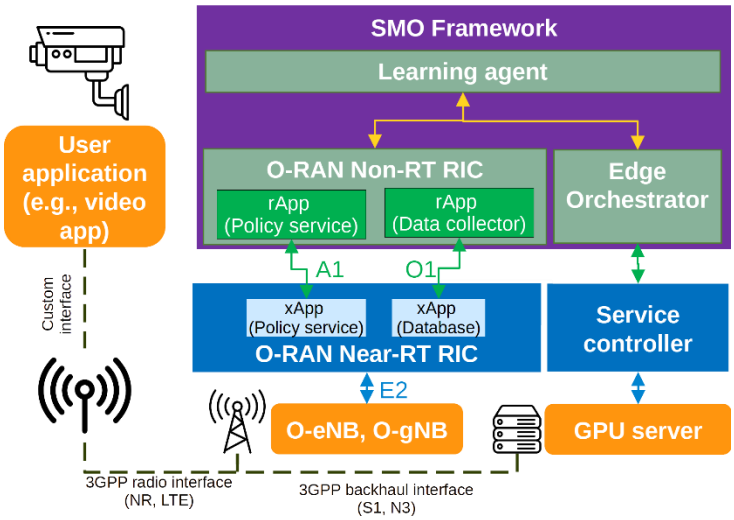


Figure 3.3.7. *O-RAN compliant system architecture.*

Contexts. We define the context at each time period t as $c_t := [\overline{c}_t, \widetilde{c}_t^{dl}] \in \mathcal{C}$, where $\overline{c}_t$ and $\widetilde{c}_t^{dl}$ are the mean and the variance of the UL CQI across all users in the slice during the previous period, and $\mathcal{C}$ is the context space.

Control policies. Our control policies were introduced in Section 3.3.1. Let $x_t := [\eta_t, a_t, \gamma_t, m_t] \in \mathcal{X}$ be the control policy selected at time period t, where $\eta_t \in \mathcal{H}$ is the image resolution, $a_t \in \mathcal{A}$ is the airtime or the uplink radio resources assigned to the slice of the service, $\gamma_t \in \Gamma$ denotes the GPU speed, $m_t \in \mathcal{M}$ denotes the upper bound of the MCS that the vBS can select to process the slice's uplink traffic, and $\mathcal{X} = \mathcal{H} \times \mathcal{A} \times \Gamma \times \mathcal{M}$ denotes the control space. We focus on UL radio policies because, as our experiments confirm, AI services like ours have little impact on the DL as the data surge goes usually upstream with only simple information (bounding boxes, labels) flowing downstream.

Performance indicators. Similarly, our performance indicators were introduced in Section 3.3.1. The service delay is denoted by $d(c,x)$, the mAP is denoted by $\rho(x)$, the consumed power at the edge server is denoted by $p^s(c,x)$, and the consumed power at the vBS is denoted by $p^b(c,x)$. Note that in practice the observations of the performance indicators are noisy (even in static setups) since the system is stochastic in nature. Remarkably, our solution intrinsically deals with noisy observation as we detail in the next section. Henceforth, we denote by $d_t(c_t,x_t)$, $\rho_t(c_t,x_t)$, $p_t^s(c_t,x_t)$, and $p_t^b(c_t,x_t)$ the noisy observations of our performance indicators at time period t. Feedback from the data plane components including all these performance metrics is received by the LA at the end of each time period t, as explained above. We assume our LA is working in a pre-production phase where the labels of the images are available for training. Alternatively, we can easily integrate other real-time precision metrics that consider the confidence output of the object recognition algorithms.

Our goal is to minimize the power consumption of the whole system (vBS and edge server) subject to performance constraints of the service. Depending on the form factor of the vBS and the configuration of the server (i.e., GPU model, motherboard, etc.) the consumed energy of each entity can have a

different order of magnitude. Moreover, the value or scarcity of the energy may also vary depending on the scenario. For instance, the energy is a scarce resource for a PoE or a solar-powered vBS. While the value of the energy at the edge may vary between day and night depending on the rates set by the power suppliers in each country. In order to capture these different scenarios, we define the following *cost function*:

$$u(c,x) = \delta_1 p^s(c,x) + \delta_2 p^b(c,x)$$

where δ_1 and δ_2 are the costs of the power at the edge server and the vBS, respectively, in monetary units per watt (mu/W). On the other hand, we consider in our work performance constraints at service-level, going a step beyond other works considering lower-level performance requirements (e.g., [17]) such as data rate or delay. The mapping between context-action pairs and the service-level performance indicators is very complex and there are no available models, as we detail in our experimental results in the previous section. For that reason, we learn them by observation. For our object recognition service, we consider two constraints: (i) a maximum service delay denoted by d^{max}, which is directly related to the frame rate (number of images per second) that the user is going to process, and (ii) a minimum mAP denoted by ρ^{min}, which indicates a lower bound on how accurate the service is in detecting the objects. We formulate the problem as follows:

$$\begin{aligned} \underset{\{x_t\}_{t=1}^{\infty}}{\text{minimize}} \quad & \lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} u(c_t, x_t) \\ \text{subject to} \quad & d_t(c_t, x_t) \leq d^{max}, \quad \forall t = 1,2,\dots \\ & \rho_t(c_t, x_t) \geq \rho^{min}, \quad \forall t = 1,2,\dots \end{aligned}$$

The nature of our problem calls for a *contextual bandit* formulation, i.e., we need to select a control policy x_t every time period t, given a context c_t. Indeed, the optimal decision at each t only depends on the current context c_t, and the performance indicators are observed at the end of each time period. Our objective is to minimize the cost in the long-term while satisfying the constraint at each time period. We would like to remark that other alternative formulations can be considered. For instance, we could consider power-constrained vBSs or an edge computing power budget by including the power consumption targets as constrains, while minimizing latency and maximizing accuracy. The flexibility of our framework allows us to implement any of these different formulations with minimal changes.

4 Capacity Forecasting

Capacity forecasting is a task in anticipatory networking that aims at reserving the resources needed to meet the upcoming traffic demand. The concept is intendedly general, and may apply to different network domain or entities, as well as to diverse definitions of demand such as aggregates or service-level ones. This functionality is especially relevant in network slicing settings, where (sets of) services run in different slices, and the operator needs to dedicate sufficient resources to each slice, in agreement with the load generated by the corresponding service(s).

In the following, we present four different use cases in the context of capacity forecasting, i.e., anticipatory allocation of resources, VM reservation in a network core datacenter, minimization of video streaming slice OPEX, and prediction-assisted network scheduling. We formalize each problem and provide a design for the solution based on diverse tools that include hybrid statistical-learning modelling, loss meta-learning, and Online Convex Optimization.

4.1 Anticipatory capacity allocation with hybrid statistical-learning models

The first capacity forecasting use-case analyzed in the project targets precisely the latter setting above, where anticipatory NI is in charge of capacity allocation to slices. Here, as in other capacity forecasting contexts, sheer accuracy is not the most relevant metric. Instead, it is critical that the capacity prediction stays above the actual load with a very high probability because underestimation determines the allocation of insufficient capacity to slices, hence service disruption on the user side. Under-provisioning also triggers violations of the SLA between the slice tenant and the network operator, which thus incurs into substantial economic penalties. Clearly, this must be avoided without allocating exceedingly large amounts of unnecessary resources, which also have a cost for the operator.

4.1.1 Problem definition

This specific problem has recently received attention, with partners of the DAEMON project having spearheaded proposals for dedicated predictors [20], [21]. These models rely on a loss function that drives the learning process to capture the actual cost of incurring SLA violations against that of overprovisioning the slice capacity. Specifically, the function handles negative and positive errors so as to reflect the different costs they entail in the context of virtualized communication networks, as follows.

- A constant penalty β is associated to each negative error, which causes an SLA violation during the predicted time interval. The parameter can be customized to the desired behavior: for instance, higher values may be used when reliability is paramount (e.g., for slices serving Ultra-Reliable Low-Latency Communications (URLLC)), and lower penalties can be applied for slices with more relaxed requirements.
- A monotonically increasing cost is attributed to positive errors, which imply the allocation of excess resources. Therefore, the cost is proportional to the amount of (unnecessarily) provisioned capacity. Typically, the expenditure is assumed to grow linearly with the overprovisioned capacity, with a fixed rate γ of cost per surplus capacity.

The configuration of the two costs can be in fact controlled by a single parameter $\alpha = \beta/\gamma$, which represents the amount of over-provisioned capacity that the operator is willing to deploy to avoid committing an SLA violation. Formally, for a given prediction error x, the loss function that abides by the specifications above is expressed as:

$$L(x) = \begin{cases} \alpha - \varepsilon \cdot x & \text{if } x \leq 0 \\ \alpha - \dfrac{x}{\varepsilon} & \text{if } 0 < x \leq \varepsilon \cdot \alpha \\ x - \varepsilon \cdot \alpha & \text{if } x > \varepsilon \cdot \alpha \end{cases}$$

where steep slopes (implemented with a small positive ε) ensure differentiability over the whole x domain [20]. The parameter α serves as a knob to steer the operational point of the system towards higher expenses in deployed resource but reduced chances of SLA violations, or vice-versa. As a result, the loss function $L(x)$ can be parametrized to the specifications of different network infrastructure locations (e.g., reflecting the higher cost of deploying resources at the network edge than at the core), resource types (e.g., capturing the fact that radio resources are sensibly more expensive than CPU resources), and SLA strategies (e.g., expressing the higher fees for violations affecting slices of critical services).

4.1.2 Proposed solution

While current state-of- the-art predictors in the networking domain, including those proposed to date for capacity forecasting, invariably rely on deep learning, very recent results from the machine learning community suggest that hybrid engines that integrate statistical modelling and Deep Neural Network (DNN) can outperform pure DNN approaches in time series forecasting tasks [22].

The aforementioned engine combines a classical Exponential Smoothing (ES) statistical model with a Recurrent Neural Network (RNN) architecture, hence is named ES-RNN [23]. It is a true hybrid predictor, since the parameters of the ES model are optimized concurrently with the RNN weights using a unified gradient descent. Thanks to this joint training, the ES-RNN model represents a leap forward with respect to previous attempts at mixing different statistical and/or machine learning methods: unlike simple combination [24] or ensemble [25] strategies used to date, this technique takes full advantage of the strengths of statistical and machine learning methods, while mitigating their respective limitations.

In the DAEMON project, we pioneer the adoption in the context of anticipatory NI solutions, and more specifically for capacity forecasting, of the hybrid statistical modelling and machine learning approach introduced by ES-RNN.

It is worth noting that the ES-RNN model is intended to operate on time series with strictly positive values of comparable magnitude. However, this assumption is often violated in the mobile networking context, where traffic is highly irregular and bursty, with continued inactivity periods that lead to a possibly significant presence of zero or near-zero values and severe underutilization of the network. We thus tailor the original ES-RNN to the specific needs of network traffic forecasting, and we introduce the Thresholded ES-RNN (TES-RNN) model. Our proposed solution adopts a threshold τ to bound the minimum value for the level used during normalization in the ES part of the model. We resort to an Automated Machine Learning, or AutoML, approach to set the value of τ in a way that is automated and does not require human intervention, via a classical Golden-Section search algorithm.

4.1.3 Sample results

We evaluate how TES-RNN can support capacity forecasting tasks 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. We hinge on real-world mobile traffic measurement data collected in an operational network to model the demands generated by the four services. We assume that each such service is assigned a dedicated network slice, and 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.

To address this problem, we train TES-RNN with the appropriate loss function proposed in [20] and also outlined above. We then compare our hybrid solution against the following three relevant benchmarks:

- INFOCOM19 [20] is the predictor designed by the study that first introduced the problem of capacity forecasting and proposed the loss function we also use to train TES-RNN. INFOCOM19 relies on a DNN architecture that receives as input a spatiotemporal representation of mobile data traffic in a 3D tensor format, and then takes advantage of convolutional layers to capture geographical correlations in the demands. INFOCOM19 is the current state-of-the-art forecasting model intended to support capacity allocation.
- ES-RNN [23] is the original ES-RNN approach. For the sake of fairness, ES-RNN is similarly trained with the capacity forecasting loss function.
- RNN is a model using the same RNN architecture as in ES-RNN and relying on a global normalization for the input data, thus without any of the optimizations proposed in TES-RNN or ES-RNN. This benchmark is useful to understand how the statistical modelling part increases the prediction accuracy. As for the previous cases, we train this benchmark with the capacity forecasting loss function.

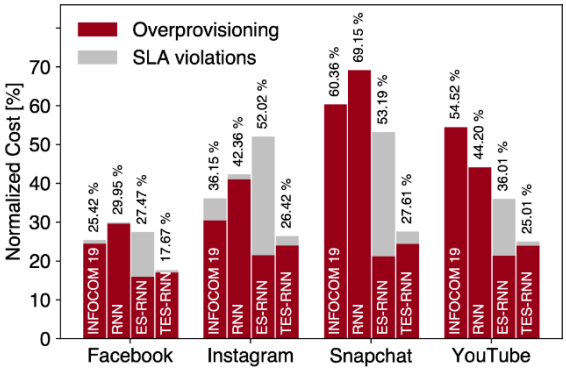


Figure 4.1.1. *Capacity allocation cost caused by INFOCOM19, RNN, ES-RNN, and TES-RNN prediction errors. Results refer to four slices at a network core datacenter, with α = 3.*

We compare the total costs incurred by the operator when using the different forecasting models to support capacity allocation. Figure **4.1.1** summarizes our main result, showing the costs determined by the four models, INFOCOM19, RNN, ES- RNN, and TES-RNN, when assigning resources to the network slice of each target service at the network core datacenter. In order to make these values interpretable, all costs are normalized to the (unavoidable) cost of the minimum resources needed to accommodate the exact demand for each service. In other words, costs are expressed as the percent excess over a baseline given by an oracle that makes a perfect prediction. The figure also tells apart the fraction of the cost resulting from the two sources of penalty, i.e., 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%. Moreover, the solution developed in DAEMON steadily guarantees very low SLA violation probabilities, which is a clearly desirable feature for the operator. Additionally, it does so by causing an overprovisioning that is lower than or comparable to that produced by the other predictors. These are very encouraging findings, as one of the benchmarks is the state-of-the-art model designed for capacity forecasting.

Interestingly, ES-RNN yields an allocation of unnecessary resources close to that of TES-RNN but incurs into much more frequent SLA violations. INFOCOM19 and RNN, when compared to TES-RNN, induce a substantial higher overprovisioning that often helps limiting SLA violations.

4.2 Virtual Machine reservation under meta-learned loss

In a second capacity forecasting use case, we consider a core network datacenter setting, where we assume that each video streaming service is assigned a dedicated Network Slice Subnet Instance (NSSI). The machine-translated intent involves that the Virtual Infrastructure Manager (VIM) responsible for controlling the datacenter resources must predict the number of VMs that need to be allocated in advance to each NSSI, so as to run the VNFs required to serve the demand generated by the corresponding mobile service. Clearly, every VM has an operating cost (e.g., due to power consumption) so it is desirable that only the strictly necessary set of VMs is reserved for each NSSI.

4.2.1 Problem definition

As we do not have access to information about the actual operating costs of the datacenter, or about the local VM management strategies, we are forced to emulate that part of the system behavior to perform our experiments. To that end, we proceed as follows: (i) we define a credible expression $f_{\mathcal{M}}$ for the target management objective; (ii) we use $f_{\mathcal{M}}$ to generate observations M_{t+1} in response to predictions d_t; and (iii) we use observations M_{t+1} as ground-truth measurements to train and test capacity forecasting models. A very important remark is that $f_{\mathcal{M}}$ is an artifice we adopt to generate measurement-like data, and it is unknown to the operator which has only access to the observations.

Formally, let all VMs have equivalent performance, so that each has operating cost κ, and can serve a maximum volume of traffic C_σ that may depend on the slice σ. Also, $\eta \gg \kappa$ is a high penalty triggered in case insufficient VMs are allocated in advance. The observations are computed as:

$$\mathcal{M}_{t+1} = f_{\mathcal{M}}(d_t, s_{t+1}) = \eta \cdot \mathcal{R}(s_{t+1} - C_\sigma \cdot \lfloor |d_t| \rfloor) + \kappa \cdot \mathcal{R}(C_\sigma \cdot \lfloor |d_t| \rfloor - s_{t+1}).$$

This expression basically forces a high handicap if insufficient VMs are allocated, and a milder one for VM overprovisioning.

4.2.2 Proposed solution

In the problem above, the main challenge is the fact that the metric $f_{\mathcal{M}}$ is complex, non-linear and non-differentiable as it includes floor and absolute value operators. Hence legacy loss functions or naïve manually designs of the loss are not effective. We propose an original approach to cope with this type of NI situations. The basic idea underpinning our design, which we name LossLeaP for Loss-Learning Predictor, is simple: instead of imposing a fixed, predefined expression of the loss function used to train a DNN predictor, we let the forecasting model free to learn the loss function that best suits the network management objective at hand. In practice, this is realized by combining a loss-learning block with the actual predictor. This second block is responsible for learning the loss function, or, equivalently, capturing the relationship between the forecast produced by the predictor and the target management objective. Once trained, the loss-learning block can operate as a tailored loss function: it receives the output of the DNN predictor and determines its quality for the precise management task. Therefore, it can be employed to train the DNN predictor so as to steer the optimization of its parameters towards minimizing the actual $f_{\mathcal{M}}$ objective.

4.2.3 Sample results

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 learn to produce a forecast d_t that minimizes $f_{\mathcal{M}}$, from observations of $\mathcal{M}_{t+1}$. We consider two benchmarks for comparison.

- 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 traffic forecasting model, implemented as the predictor part of LossLeaP, and trained to minimize a Mean Square Error (MSE) loss. Note that this model requires an additional downstream block in charge of taking the management decision. In this use case, we manually design a VM allocation algorithm that (i) applies an overprovisioning factor to the predicted traffic so as to avoid expensive under-dimensioning, and (ii) uses C_σ to compute the number of VMs to serve the inflated demand. Upon extensive tests, we set the overprovisioning factor to 1.6, which enforeces 60% safety margin.

Figure **4.1.1** summarizes the performance, when the C_σ and κ are set to 10% and 1% of η. 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.

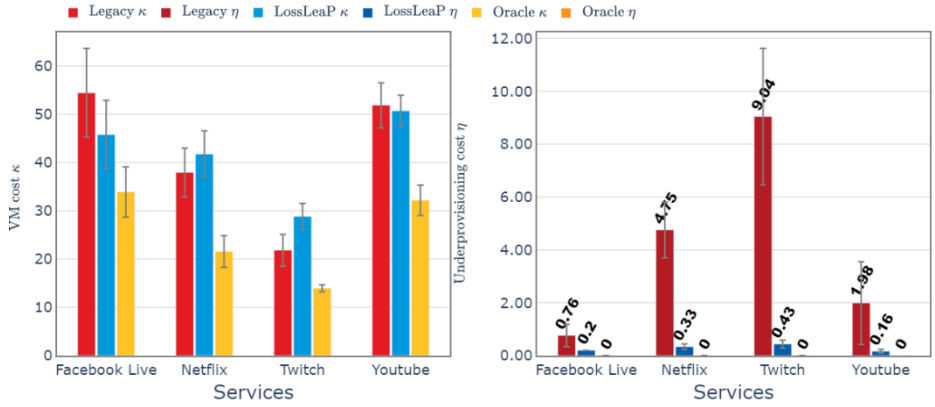


Figure 4.2.1. *VM reservation for diverse slices. (a) Reserved VMs. (b) Fraction of time when the slice demand cannot be served.*

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

The third use-case for capacity forecasting considered in the project is that of a mobile Edge environment, where a set of computational facilities, each serving between 70 and 130 BSs, are located close to the radio access. We assume again that each of the four video streaming services has a dedicated NSSI at the Edge facilities. Here, the management intent aims at 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.

4.3.1 Problem definition

As for the previous use case, we have to emulate the ground-truth OPEX, by developing a sensible model that relates OPEX to the system variables. Here, we go a step further in terms of complexity and, rather than defining a single expression for the objective, we employ a whole pipeline to mimic the objective. Figure **4.3.1** offers a high-level view of the pipeline. First, the slice-level allocated computing capacity d_t and the future traffic to be served are split across users according to a probability distribution that describes the imbalance of traffic generated by different users.

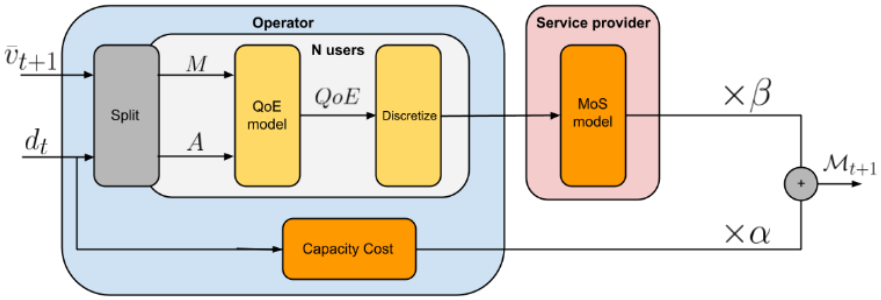


Figure 4.3.1. *Emulation pipeline for the monetary OPEX.*

The per-user maximum requested, and available powers are fed to an empirical non-linear model that maps such metrics into a user-level video streaming QoE [26]. The QoE value is then discretized in line with real-world QoE metrics and passed to a module that converts the QoE into a Mean Opinion Score (MOS). Based on the MOSs of the users, the slice tenant can communicate to the operator if the SLA has been violated, which entails a monetary fee β per violation. This cost is summed to that generated by the need to reserve the compute resources d_t to accommodate the slice traffic, derived in the bottom block, at a price a per capacity unit.

Interestingly, not only the expression of the objective is now much more involved than before, but it is also not observable by the operator in practical cases. Indeed, the VNF Manager (VNFM) that must take the anticipatory decision only has data about the current allocated resources and slice traffic demand from the VIM. In the best case, it may consume QoE information provided by the Network Data Analytics Function (NWDAF) [27] that interfaces with Application Functions (AF), but the MOS part of the pipeline remains unknown. Ultimately, this use case sets forth a scenario where a network manager would not have the necessary system knowledge to manually design a solution to the problem.

4.3.2 Proposed solution

The LossLeaP model introduced in Section 4.2 can be employed to address the challenges of this specitic use case as well. More precisely, we let LossLeaP learn the whole pipeline of the objective in a setting where the compute capacity reconfiguration occurs at every 5 minutes. We compare it against two benchmarks, as follows.

- An Oracle predictor that knows the future demand and the full objective pipeline and returns an optimal allocation.
- DeepCog, a state-of-the-art capacity predictor [20] that we configure to cope with the problem in the best way possible, by setting its loss function parameter to β/α.

4.3.3 Sample results

The performance for a slice dedicated to Facebook Live traffic is summarized in Figure **4.3.2**. All costs are relative to that of the minimum static capacity allocation that always services the full demand and are shown for different β/α ratios. Despite the fact that we feed it with information about the true a and β values, DeepCog is still constrained by its inflexible loss function. Instead, LossLeaP can autonomously learn a much better loss, which results in a reduced cost closer to the Oracle one.

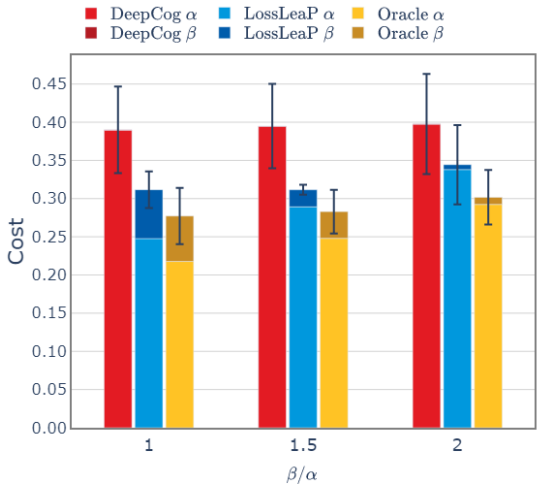


Figure 4.3.2. *Overall OPEX cost in the Facebook Live slice case.*

4.4 Prediction-assisted network scheduling for data analytics

As a final step in this task, we focus on algorithms that leverage predictions in order to implement a special class of services, namely edge analytic services. Indeed, today a growing number of mobile network services require the execution of some type of Machine Learning (or analytic) tasks in proximity with users. Prominent examples include mobile gaming and AR applications [28], cognitive assistance and remote control of robots [29]. In these services the network collects, and processes data streams emanating from (small) devices and returns the results to users in real-time. For example, in MVA, handheld devices capture and transmit images to servers which classify depicted objects in milliseconds [30] [31]. However, these *edge or analytic* services raise previously unseen resource management challenges for the network.

4.4.1 Problem definition

First, unlike cloud-based centralized services, deployments at the edge receive relatively few requests with properties (e.g., arrival patterns) that vary substantially across space and time [32]. Therefore, the expected request load is typically non-stationary and cannot be a priori modeled. This issue is exacerbated by the heterogeneity of user devices where some assist the service by data pre-processing, others have tight energy budgets restraining their transmissions, and so on. Secondly, the analytics performance cannot be quantified using generic metrics such as the amount of processed data [33] but should be measured in terms of accuracy, granularity, or confidence. Thirdly, these metrics depend on the end-to-end processing pipeline, e.g., the image encoding used by the devices, the DNN size that is selected by the server to process the images, etc. Importantly, the effect of these decisions is initially unknown, e.g., the DNN impact on accuracy is revealed only after using it.

4.4.2 Proposed solution

Our starting point will be to build on the theory of Online Convex Optimization (OCO) which we will extend in two important ways: include time-average (or, budget) constraints that are very important and common for communication systems; and include predictions for the future network state and user requests. OCO-based algorithms have been very successful in ensuring a minimum performance under any possible scenario, i.e., even under adversarial network conditions. However, their performance is often very pessimistic or conservative. At the other extreme, the wide availability of neural networks and other AI/ML tools has spurred the development of network control algorithms that use these tools. While their success is impressive, it is common ground that when the training data do not fit the actual scenario or when there are sudden changes in the users' profiles (system resources, service requirements, etc.), then making scheduling decisions using such predetermined function approximators can be particularly ineffective.

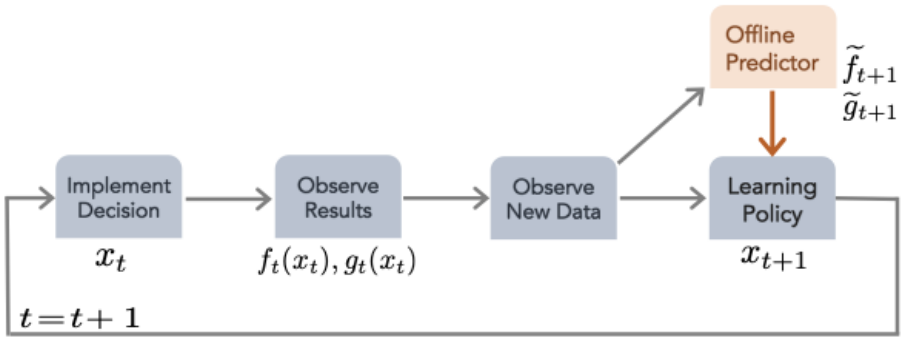


Figure 4.4.1. *The blueprint algorithm for incorporating predictions in online learning.*

Figure **4.4.1** depicts schematically the high-level idea of using predictions, that are made available via an offline predictor (e.g., a DNN) to the online learning policy. The envisioned algorithm operates in an iterative fashion where the network scheduling decisions are taken without knowing the next-slot system state but with the help of a prediction. When the results of the decisions are observed, the learner updates its beliefs and using the next-slot predictions makes a renewed scheduling decision. The performance of such algorithms is often measured in terms of the regret, i.e., the distance of the achieved overall performance from an ideal, unknown, optimal scheduling policy that we could have devised if we knew a priori the entire sample path. During the course of DAEMON we plan to pursue this thread and develop a suite of such algorithms that can serve as independent NI components in B5G and 6G networks.

4.5 Summary

The use cases above highlight several capacity forecasting problems where models based on machine learning and optimization offer substantial and clear opportunities for effective automation. Whenever the design stage of the solution was advanced enough, we carried out extensive tests in realistic settings and against a wide range of benchmarks, reporting above a representative subset of the results that demonstrate the viability of our concept, as well as the potential performance gains it can unlock. Overall, the work carried out to date by the DAEMON project contributes to advancing the state of the art in automated network management.

5 Automated anomaly response

Anomalies in the network operation due to misconfigurations, malfunctions, or malicious behaviors risk to represent a particularly serious complication in B5G systems whose management will be already characterized by substantial added complexity with respect to 5G infrastructures. In current human-centric anomaly detection and resolution processes, whenever a failure occurs, engineers must sift through large amounts of system logs to understand what might have gone wrong and which were the customers affected; this is often performed by inspecting the logs manually with keyword search and rule matching. Thus, there is a clear need to align anomaly handling techniques with the automated approaches that will permeate all other B5G management procedures.

To answer this need, our goal is to build specific NI solutions to automatically detect unexpected behaviors that emerge in the network and trigger relevant signals. *The NI-native architecture proposed by DAEMON is the perfect environment to assist the development of a NI that is dedicated to anomaly management*: the interaction among NI instances located in different network domains enables developing better-informed anomaly detection algorithms that feed on data collected simultaneously across all B5G micro-domains – and possibly preprocessed by the local NI instances according to a FL model to minimize anomaly discovery times. *On top of the baseline anomaly detection functionalities, DAEMON will ensure that the dedicated NI addresses the whole lifecycle of the anomaly management process and takes care of subsequent root cause analysis and automated response*; this reduces reaction times, mitigates the impact of anomalies on the network operation, and improves reliability.

5.1 Federated learning powered anomaly detection in B5G/6G

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.

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 the "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" (e.g., references below [34] [35]). The problem has attracted attention in various domains and for various solution approaches [36]. 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.

Our architecture will follow the FL approach [37], 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 **5.1.1** 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.

Based on the scoping so far, the next steps (to be reported in the next deliverable, D4.2) concern the development of models, the realization of validation activities (result collection and analysis), and the support of dissemination and demonstration activities.

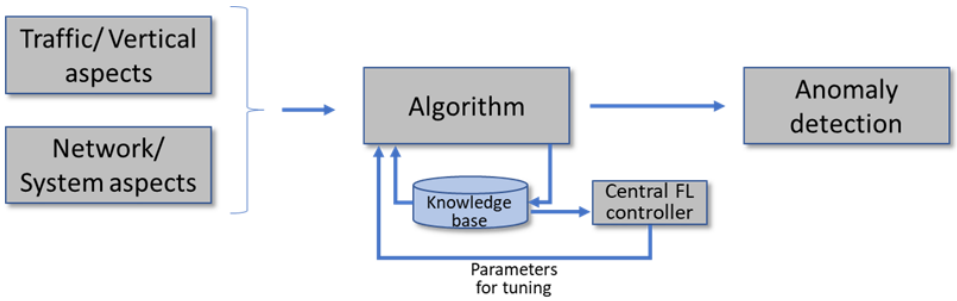


Figure 5.1.1. *Overview of our approach.*

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

In the last few years, we have been witnessing a growing popularity of IoT and Machine to Machine (M2M) applications from connected cars to logistic and wearables. These applications rely on traditional mobile network infrastructures and on a complex ecosystem composed by several actors. Indeed, IoT customers (e.g., car manufacturers, delivery services) contract connectivity and added value services (e.g., security, management, Application Programming Interfaces (APIs)) from IoT/M2M platforms. These platforms rely global Subscriber Identity Module (SIMs) cards to offer worldwide connectivity, i.e., SIMs provided by a given MNO that exploits roaming to provide connectivity in foreign countries. While some global SIM are used in the origin country of the provider MNO, the vast majority of them (75%) operates in roaming [38]. In turn, MNOs mostly rely on third party providers, the Internet Protocol (IP) Exchange (IPX) providers [39], to offer roaming services to global IoT SIMs.

The usage of global IoT global SIMs allows IoT platforms and customers to avoid the cost of establishing technical and commercial relationships with a local operator in every country they operate. Also, it enables centralized control and management of the IoT applications. However, guaranteeing communication availability and reliability to IoT verticals is already hard when only one commercial operator is involved; this is even more challenging in the case of global SIMs, when multiple parties and platforms are involved, and only have a partial view of the communication (i.e., IoT customers, M2M platforms, IPX providers and MNOs). In addition, while most platforms have ticketing and monitoring systems in place, operations are mostly reactive in dealing with communication issues. Consequently, many incidents become severe with customer escalation/reporting or large platform impact, and the service is often compromised.

In this deliverable, we present our experience in designing, building and deploying an anomaly detection solution to proactively detect service availability and reliability issues of IoT customers of a large commercial IPX provider. The goal of the solution is twofold. First, to leverage the IPX broad visibility on network behavior of IoT devices. Indeed, IPXs are at the core of the communication infrastructure and, while issues can happen at different locations, the IPX provider is potentially able to detect anomalous behavior and identify the source of the issue. Second, to identify anomalous behavior (e.g., aggressive data generation from the IoT application at the end device, coverage problems in the visited network where the IoT device is deployed) as they appear across different IoT verticals and customers with different requirements. This enables to address incidents before they become critical and impact services and platforms.

The contribution we include in this deliverable are as follows:

- First, we first describe the IoT service, infrastructure, and network visibility of the IPX provider we monitor. Our IPX provider is a Tier-1 Internet Service Provider providing services to millions of IoT devices, with 100+ POPs in more than 40 countries.
- Second, we describe the main challenges and operational requirements of a proactive anomaly detection system for this IPX.
- Third, following the requirements we identify, we propose three deep learning models for IoT verticals anomaly detection based on passive signaling traffic. We describe our experience in the design and deployment of our solution with the operational team.

5.2.1 Problem Formulation

The main goal of our solution is to enable operation teams to proactively identify communication issues before they become critical, and the service is compromised. This departs from current operational approaches, where operation teams reactively deal with incidents when the issue is severe, and the service – likely compromised. In this section, we describe the main requirements and challenges we identified based on our experience building the anomaly detection solution.

Input Data Availability. Because of operational constraints, the operation team does not collect information related to the data roaming service for all devices and locations, but rather for a limited subset of them. Therefore, we design the anomaly detection solution to leverage mobility signaling data only. We find that there is a strong correlation between signaling and data roaming traffic of every device which would allow us to design a useful anomaly detection solution even with this partial information.

Customizable per Vertical Solution. The traffic patterns of IoT devices vary across end customers, as well as within devices of a given customer [40]. The IPX does not have information about the specific applications the devices serve, either because it does not have direct relation with end customers or because this information is not provided. It follows the solution needs to be applied in a customized fashion to each group of devices with similar traffic patterns, without assuming prior knowledge about the customer or final purpose of the IoT devices. We follow a two steps approach. First, we identify groups of devices per customer having similar traffic pattern via clustering. Second, we an anomaly detection design on each cluster.

Incidents and Anomalies. The operation teams leverage a monitoring system with rule-based alarms triggers. These rules are derived from their operational experience in dealing with major incidences either impacting customer services or the IPX platform itself. These incidents are also registered in the ticketing system with some description of the issue and devices impacted. On the one hand, this constitutes a source of ground truth to design and validate the anomaly detection design. On the other hand, as the operations mostly deal with critical incidents only, many (potentially harmful) anomalous patterns may have never been registered, and are of interest to be identified by the operation teams. Our design has therefore to identify both known and unknown incidents. We leverage an unsupervised deep learning approach to learn the normal behavior of devices and identify deviations from these normal patterns (i.e., anomalies).

Usability and Explainability of the Output. The anomalies identified by our detection engine may be real incidents or simply deviations from "normal" behavior without being harmful. It is critical to prioritize and provide a tool to the operation teams to easily identify incidents to investigate and what are potential causes of the issue. We then follow three approaches to validate, tune the design and provide a useful output to operation teams. First, we use the information available in the ticketing system to understand if the model is able to detect known incidents. Second, we validate the anomalies generated by the solution against raw data to understand the cause for which an anomaly has been triggered. Third, we jointly work with the operation teams to distinguish between anomalies that are actual incidents, and non-harmful anomalies.

Operational Requirements. The proactive anomaly detection solution must integrate with the existing monitoring and analytics infrastructure of the IPX. As the platform provides data collection and frameworks to execute machine learning pipelines, we can easily integrate our design in the existing infrastructure without major modifications. There are nevertheless two important aspects to consider for its deployment. First, it is critical to define at which timescales the engine must be used (e.g., hourly, daily). As data is collected every 5 minutes by the data collector, the anomaly detection engine could run at this timescale. Nevertheless, as major incidents are already detected by the reactive rule-based alarm triggers, the operation team indicated us that running the model once a day is enough for them to proactively identify incidents that may become critical and investigate them. Second, we need to define the time validity of the model (i.e., how often to retrain the model).

5.2.1.1 Monitored System and Dataset Overview

Communication of IoT devices and applications is today provided via a complex ecosystem composed of several actors often belonging to different organizations. This ecosystem, pictorially represented in Figure **5.2.1**, can be divided in IoT logical layer and communication infrastructure.

On the *IoT logical layer*, IoT customers (e.g., car manufacturer, delivery services, antitheft systems) contract services from M2M/IoT platforms. These platforms provide their customers connectivity via a set of SIMs that can be installed in IoT devices and used globally (i.e., global SIMs). They also provide management in a centralized manner and other added values services (e.g., security, APIs) on the SIMs delivered to every customer.

However, M2M platforms do not provide themselves the *communication infrastructure* but rather contract the global SIMs from one or multiple MNOs. These MNOs are therefore the ones providing connectivity via their infrastructures. Also, as most of the IoT devices are used outside the origin MNO country, MNOs must provide roaming services to guarantee worldwide connectivity to global SIMs. This is usually done via third party providers - the IPX providers - that peer among them to offer global world-wide coverage and interconnect about 800 MNOs worldwide. IPX providers often provide dedicated services for IoT traffic, mostly consisting of data roaming and signaling for IoT devices. Note that some MNOs uses the IPX IoT service also for SIMs operating in their origin countries (e.g., Virtual MNOs), MNOs using the IPX for General Packet Radio Service (GPRS) Tunneling Protocol (GTP) tunnel management.

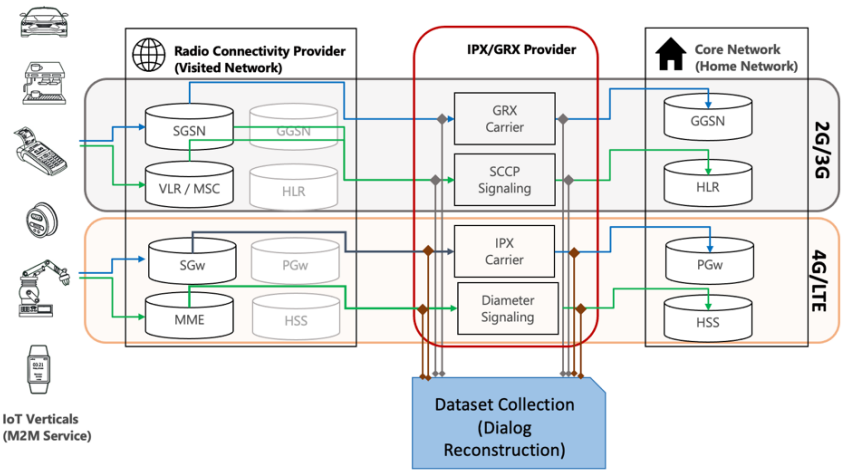


Figure 5.2.1. *High level architecture of the IPX-P's monitoring to build our dataset. We build our dataset using a commercial software solution that processes the raw signaling traffic (SCCP, Diameter or GTP), and that rebuilds the dialogues between the different core network elements. We build datasets for 2G/3G as well as 4G/LTE.*

In this study, we focus on the M2M service provided of a large operational IPX provider with more than 100 PoPs in 40+ countries with a particularly strong presence in Europe and America. In the following, we describe the M2M service of the IPX provider, its infrastructure and visibility in IoT customers' traffic.

We monitor the IPX provider 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 **5.2.1** 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 3 summarizes the datasets we use to characterize the operations of an IPX provider with a large international footprint.

Table 3. *IPX Datasets.*

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 re-quired 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.

5.2.1.2 *Dataset Exploration.*

In the following paragraphs we focus on the traffic corresponding to the IoT devices that the IPX-P M2M platform connects for two weeks period in December 2019. Figure **5.2.2** and Figure **5.2.3** show the time series of average number of signaling messages per device for 2G/3G (Figure **5.2.2**) and 4G/LTE (Figure **5.2.3**), as well as the 95th percentile calculated over one-hour intervals. To put these results in context, we include statistics from a similar number of smartphones using the same radio technology. We selected the set of smartphones leveraging the device brand information, which we retrieve by checking the International Mobile Equipment Identity (IMEI) and the corresponding Type Allocation Code (TAC) code and included only iPhone and Samsung Galaxy devices (the two most popular smartphones) in the pool. Figure **5.2.2** and Figure **5.2.3** show that IoT devices generally trigger a higher load on the signaling infrastructure, regardless of the infrastructure they use (Diameter or Signaling System 7 (SS7)). This holds when checking either the average number of messages per device across time, or the 95% percentile per hour across all devices.

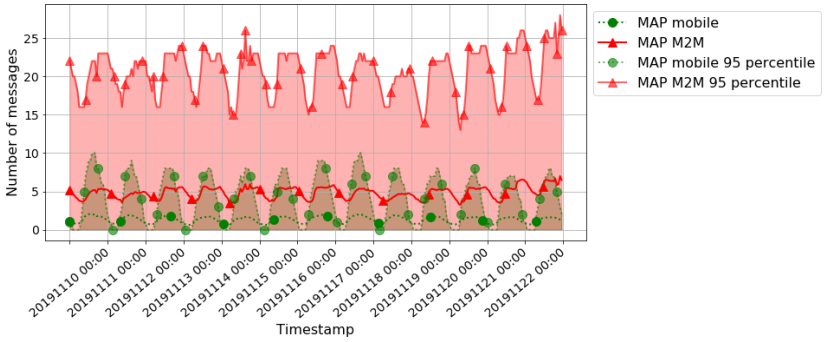


Figure 5.2.2. *2G/3G M2M/IoT devices and smartphone devices.*

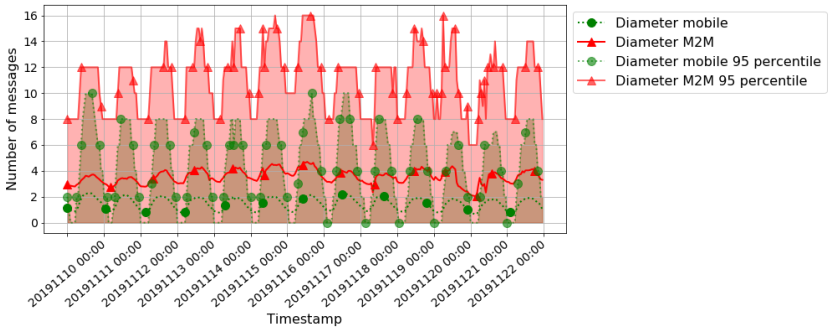


Figure 5.2.3. *4G/LTE M2M/IoT devices and smartphone devices.*

We also compare the duration of roaming sessions (i.e., the total number of days a device sent at least one signaling message while in roaming) for both IoT devices and smart phones.

Figure **5.2.4** shows the number of days an IoT device was active during the two-week period we analyze. We note that the majority of IoT devices have long roaming sessions, which in our case cover the entire observation period. This is very different to what we observe for smartphones (left), whose roaming session lengths are shorter. This is expected, since IoT devices are meant to provide services in the country where the IoT provider deploys its services during long periods of time, thus becoming "permanent roamers" in the visited country. At the same time, this also translates into significant signaling load on the Virtual Mobile Network Operator (VMNO) infrastructure from inbound roaming IoT devices.

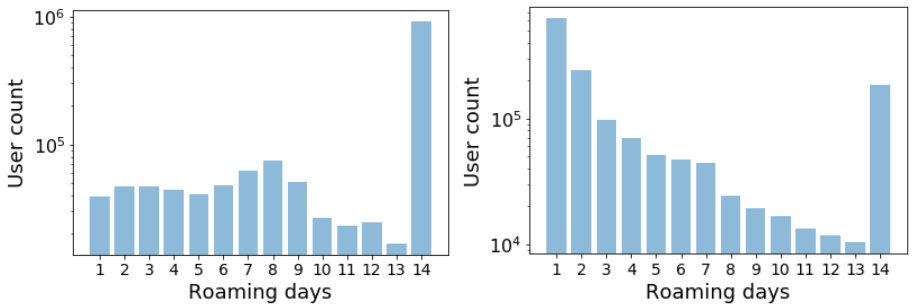


Figure 5.2.4. *Roaming session duration for IoT devices (left) and smartphones (right).*

5.2.2 Proposed solution and (initial) evaluation

We propose an unsupervised method for anomaly detection of IoT devices, agnostic to the device usage. The key idea of our model is to learn the normal network behavior by estimating a low dimension representation of the signaling messages exchanged by IoT devices. Our methodology consists of three major steps (Figure **5.2.5**), which we detail next.

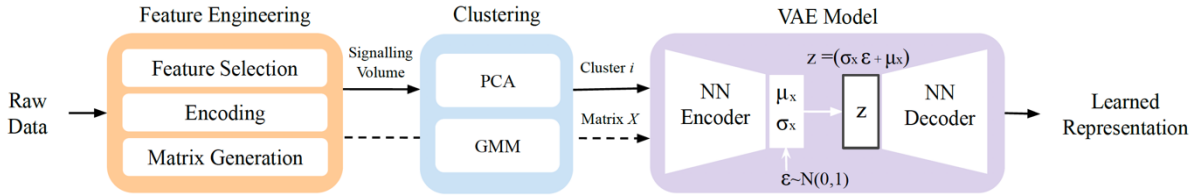


Figure 5.2.5. *The proposed anomaly detection framework.*

First, we transform the raw data into effective features for the anomaly detection approach. This includes cleaning, encoding, and generating the input data matrix for the Variational Autoencoder (VAE) model (see the "Feature Engineering" block in Figure **5.2.5**)).

Then, we cluster the devices by using the number of messages per day. We noticed that sub-groups of devices have very different signaling patterns. Given that our dataset is agnostic to the device purpose (i.e., the IoT vertical), we propose an unsupervised clustering approach for uncovering the normal signaling trends within our dataset. This step is crucial for obtaining a good performance model, otherwise if we trained the model with all devices, it would detect the minority class of devices as anomalies because of its different patterns.

For each cluster we apply three anomaly detection models to be compared among them (see "Models" block in Figure **5.2.5**). Our models are a Gaussian Mixture Model (GMM) as baseline and a VAE.

The approach we put forward relies on a likelihood-based model (i.e., bow-tie architecture VAE), which consumes the encoded matrices for learning a low dimension representation (z in Figure **5.2.5**) of each group of devices. We train the model so that the latent representation effectively represents the normal behavior of devices. At inference stage, we compute an anomaly score for each of the devices by comparing new representations to the learned one. This offers the opportunity to detect anomalies caused by different unknown reasons, indistinct to the device purpose and without requiring any labeled data.

5.2.2.1 Data Representation.

We perform feature engineering to transform raw data into effective features for the model. Our dataset is composed of Diameter and MAP records which contain 96 and 48 fields, respectively. We employ the domain knowledge of the operations team to define the fields and the encoding that best exploits the data for anomaly detection. The fields we decide to keep are: International Mobile Subscriber Identity (IMSI), start date and time, duration, violation, request type (operation code in MAP) and result code (error code in MAP).

We encode the features by applying the mapping shown in Table 4. We bin the values of "duration" by defining ranges that differentiate between normal and unusual values, so that the model can discriminate easily between these values. In this case, a duration larger than 2.5 sec would be considered as unusual, while being above 6 sec would be very rare. "_" refers to null values. "Violation" indicates whether the sequence contains all messages or whether some error occurred.

Table 4. *Feature encoding scheme.*

Fields	Values	Encoding	Row
Duration [ms]	"_"	0	0
	0 ≤ value ≤ 2,500	1	1
	2,500 ≤ value ≤ 6,000	2	11
	6,000 ≤ value	3	12
Format Violation	0	0	2
	≥ 1	1	13
Result Code	"_"	0	3
	1009 (Informational)	1	4
	2001 (Success)	2	5
	3*** (Error)	3	14

	4*** (Transient Failures)	4	15
	5*** (Permanent Failure)	5	16
Request Type	"_"	0	6
	Update Location	1	7
	Cancel Location	2	8
	Auth-Information	3	9
	Notify	4	10
	Purge UE	5	17
	Insert Subscriber Data	6	18
	Delete Subscriber Data	7	19
	Capabilities Exchange	8	20
	Re-Auth	8	21
	Other	9	22

We encode it as a binary variable, not considering the type of violation occurred. We bin the values of "Result Code" based on the first value of the code, as it indicates whether a particular request was completed successfully or whether an error occurred. Finally, we map different integer values to the most frequent "Request Type" codes and assign the same category to the rest, as they appear very few times.

We noticed that a subset of the devices only presented signaling records for few days. To guarantee that meaningful statistics are captured in the training dataset, we perform a device filtering operation, discarding a device if it is present in the network less than 25% of time. We set this threshold empirically, which discards around 13% of the devices, and still retaining tens of thousands of devices.

As the main objective for anomaly detection is intervention on the anomalous devices, and such interventions cannot be typically made in "real-time" by operators, we focus on providing the anomalies with daily resolution. However, the input data do not need to be at the same resolution. Therefore, we will work with 15-minute intervals, so we gain information from the fluctuations during the day. Having the data encoded, we group records by "IMSI" and generate daily matrices by counting the number of occurrences of each value every 15 minutes, resulting in matrices of $D \times N$. Being the rows the values of the fields (D=36), the columns the timescale granularity (N=96), and every entry a counter. This representation of the data enables the VAE model to learn the distribution of signaling data per IMSI and per day.

5.2.2.2 Clustering

The common approach in ML is to build one model over the entire dataset. However, as we previously described, we have no information into the use of the IoT devices, and we see very different signaling patterns in our data. Therefore, we implement an unsupervised approach to group devices with similar signaling traffic volumes before ingesting them into the anomaly detection model.

We use the number of MAP and Diameter messages per device and day for clustering. This feature enables to distinguish between high or low signaling devices and MAP or Diameter connected devices, which gives us enough information to divide them in different kind of behaviors. Our clustering model is a GMM with as many components as days in the training dataset (i.e., 30 components if we use a month of data. We choose the number of clusters based on its homogeneity and volume to avoid very specific clusters having few samples.

We generate the clusters using the training data. At inference, we use the previous month of data to assign devices to the generated clusters using the Nearest Neighbor Algorithm, which classifies new device based on how similar devices (its neighbors) are classified.

5.2.2.3 Anomaly Detection Models.

We are evaluating five different unsupervised models on our data: GMMs, Variational Autoencoder (VAE) with Fully Connected neural networks (FCNNs), VAE with Convolutional Neural Networks (CNNs), VAE with Recurrent Neural Networks and Sequence to Sequence (Seq2Seq). Anomalies are unknown, as we found that it is not feasible to train a detection model in a supervised manner (i.e., we do not have an exhaustive list of possible anomalies that can occur in the system).

PCA-GMM. The GMM assumes that regular data is generated from a mixture of Gaussian distributions with unknown parameters. The parameters of the distributions are fit using the expectation-maximization algorithm which maximizes the likelihood of the data points. We assign to each data point a probability of being generated by the distribution, if the probability is very low that is an indication of being an anomaly. We aim to use this model as our baseline.

VAE. The Variational Autoencoder consist of two sub-networks, namely an encoder and a decoder. This architecture uses a bottleneck to learn a low-dimensional representation space z of the training data x. The encoder learns the mapping of the input data x into a prior distribution $p(z)$, which is usually a standard Gaussian $z \sim N(0, I)$. The last layer of the encoder outputs the parameters μ_x, σ_x of the distribution $q(z|x)$. The decoder network reconstructs $x \sim p(x|z)$ from the latent representation z sampled from $N(\mu_x, \sigma_x)$. In contrast with the GMM, this model does not assume that data comes from a Gaussian distribution, but it learns the transformation of the data distribution into a Gaussian. To train a VAE, we need two losses, one is Kullback–Leibler (KL) divergence loss $L_{KL,}$ which is used to make sure that the latent vector z is an independent unit Gaussian random variable. The other is the reconstruction loss L_{rec} which measures how accurately the network reconstructed the input matrices.

There is a trade-off between how accurate our network can be and how close its latent variables can match the unit Gaussian distribution. Higher β gives more structured latent space at the cost of poorer reconstruction, while lower values give better reconstruction with less structured latent space. We set β to 1, giving the same weight to both loss functions.

We develop three VAE models containing different neural networks (FCNN, RNN and CNN) to compare its performance. We made the models to share the following hyperparameters for a fair comparison: a latent space dimension of 30, batch size of 32, learning rate of 10^{-4} and ADAptive Moment estimation (ADAM) optimization, which enables training in minibatches.

- ***FC-VAE.*** Our simplest approach is the FC-VAE, which both encoder and decoder networks are composed of 4 fully connected layers with Rectified Linear Unit (ReLU) activation functions. The encoder reduces the number of features up to z=30 (512×512×256×30) while the decoder does the reverse process, in-creasing the number of features at each layer (30×256×512×512).
- ***CNN-VAE.*** This model contains 4 convolutional layers with different kernel sizes ([25, 3], [3, 12], [5, 5], [3, 3]) and astride fixed to 1, in both encoder and decoder networks. We use vertical filters to learn the relation between fields at same timestamps, while horizontal filters are used to learn the time-variant features. Each convolutional layer is followed by a batch normalization layer [9] to help stabilize training and a ReLU activation layer. In the encoder, the CNN layers are followed by two fully connected output layers (for mean and variance), used to compute the KL divergence loss and sample latent variable z. In the decoder, for up-sampling we apply a 2D transposed convolution operator with padding on the encoded data.
- ***LSTM-VAE.*** In this model we use a bidirectional Long Short-Term Memory (LSTM) layer of 512 in the encoder followed by two fully connected layers to compute the statistics of the latent space, as explained above. The decoder recon-structs the input by using a fully connected layer followed by a bidirectional LSTM of the same size.

5.2.2.4 *Initial evaluation*

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 **5.2.6**, 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.

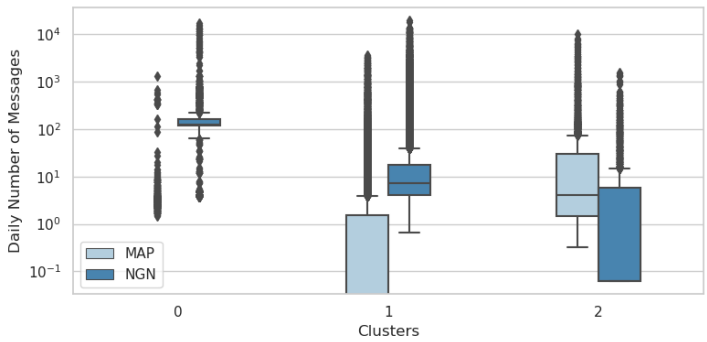


Figure 5.2.6. *Clusters of devices: We find three groups of devices containing different amount of signaling traffic each. Groups are well defined as the upper and lower quartiles of the boxplots do not overlap between them in the vertical axis.*

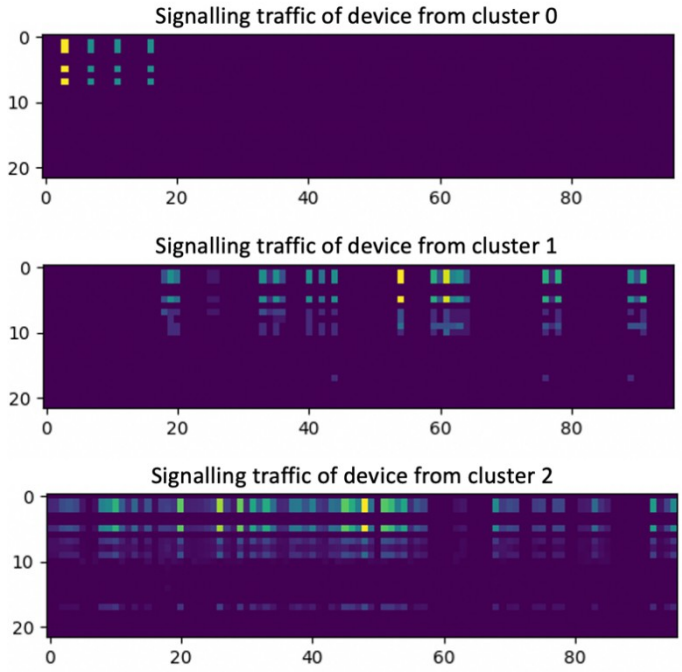


Figure 5.2.7. *Example of signaling data representation for devices from different clusters.*

We show in Figure **5.2.7** an example of the data representation for one (randomly chosen) device from each of the three clusters we obtain. The visual difference between the signaling patterns is obvious.

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.

To evaluate our models, we aim to build 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 though incidences but missed anomalies that were not registered because nobody reported them. Thus, we aim to work to validate the detected anomalies with the operations team and compute classical information retrieval measures, such as precision-recall curves and the average precision measure.

6 Self-learning MANO

In the era of B5G and 6G, a key innovation compared to its predecessor is the introduction of the virtualization technique, the well-known NFV, which allows the network infrastructure to be shared among different stakeholders by creating virtual instantiations of physical resources, in the form of VNFs. However, the virtualization of physical resources comes hand-in-hand with increased network management overhead, due to the additional relations between physical and virtualized resources.

Moreover, thanks to NFV characteristics, different slices of the network can be straightforwardly created to serve multiple tenants, compared to PNFs heavily relying on dedicated hardware, and thus fulfill specific network slice requirements. Nevertheless, one shall notice that human operators cannot keep track of all VNs, given the diverse demands of NSs. Therefore, automatic solutions are needed in the management of NFV-based networks that are flexible enough to adapt to diverse working conditions without human intervention. In this regard, the ETSI has defined several MANO operations that are critical for the correct operation of NFV-based networks [41], and scaling is one of them. To work properly, these VNFs require a minimum amount of resources, e.g., processing power in terms of CPUs, which indicates the number of jobs per second it can process. Moreover, these VNFs can be deployed in Commercial Off-The-Shelf (COTS) servers with limited CPU capacity, and as a consequence, the number of VNF instances is limited as well. The auto-scaling problem can be defined as how to dynamically add or remove VNF instances to serve a variable workload [42]. This workload must be served under a given SLO, e.g., maximum service latency. To meet this SLO, the amount of computing resources must be dynamically and efficiently changed, without incurring in under- or over-provisioning. In short, we put particular focus in this section on the Self-Learning MANO use case [1] on the resource scaling over the edge microdomain.

In this section, we study four topics to enable self-learning MANO. First, we consider the issue of admitting new requests by taking placement/routing into account at the set-up time of a request. Second, the (horizontal and vertical) scaling decision is investigated at instants of which the monitoring function indicates some (imminent) problems, which may come from traffic prediction or anomaly decision NI functionality introduced in the prior section. Thirdly, the economical aspect is considered in terms of leasing cost when realizing MANO operation with re-scalable infrastructure. In the last part, we cover the support of diverse AI algorithms to enforce decisions and conduct monitoring and diagnostics regarding the efficiency of the decisions.

6.1 Placement and routing new requests

In this subsection, our aim is to deal with the admission control of new Service Function Chain (SFC) requests, being composed of interconnected VNFs, and the subsequent placement of their associated VNFs on COTS servers and the routing of their associated flows over the network connecting these servers. Both SFC requests and underlying infrastructures can be modeled as directed graphs and our goal is to apply NI solutions to improve resource utilization and decrease rejection ratio.

6.1.1 AI-enhanced MANO in B5G/6G

As an extra step an orchestrator, e.g., following also the ETSI Open Source MANO (OSM) approach, with AI embedded capabilities will enable advanced automation and intelligence to better manage and take actions when necessary based on the specific use case requirements. An interface from the orchestrator to the verticals will be created in order to interact with each other and transfer information. An interface from the orchestrator to Verticals will be created in order to interact with each other and transfer information. The orchestrator receives requirements per application and use case from the verticals to create the optimal deployment of VNFs as illustrated in Figure **6.1.1**.

The algorithms take as input the provided information for the different nodes of the network to reach optimal decisions. These algorithms are part of the embedded AI of the orchestrator that enable the intelligence and automation of the system, through optimal placement and resource utilization. After the decision for deployment based on the current status of the infrastructure is reached, the vertical's VNFs are deployed and initialized. When deployed, network and application services will continue to be orchestrated and managed by the orchestrator and be available to the customers without changing any of the existing business services.

Metrics from deployed VNFs will be collected and reported back to the orchestrator in order to re-evaluate the allocation and deployment of the VNFs to always be compliant with the requirements provided by verticals and infrastructure. The orchestrator's embedded AI will continue to analyze the data collected, historical information that could be provided from verticals and other data from infrastructure conditions to find out if changes to the deployment of VNFs need to be done. If a metric does not meet the vertical requirement for a specific VNF, the orchestrator will make the necessary changes to fix it, either by relocation of the VNF, scaling or other appropriate actions.

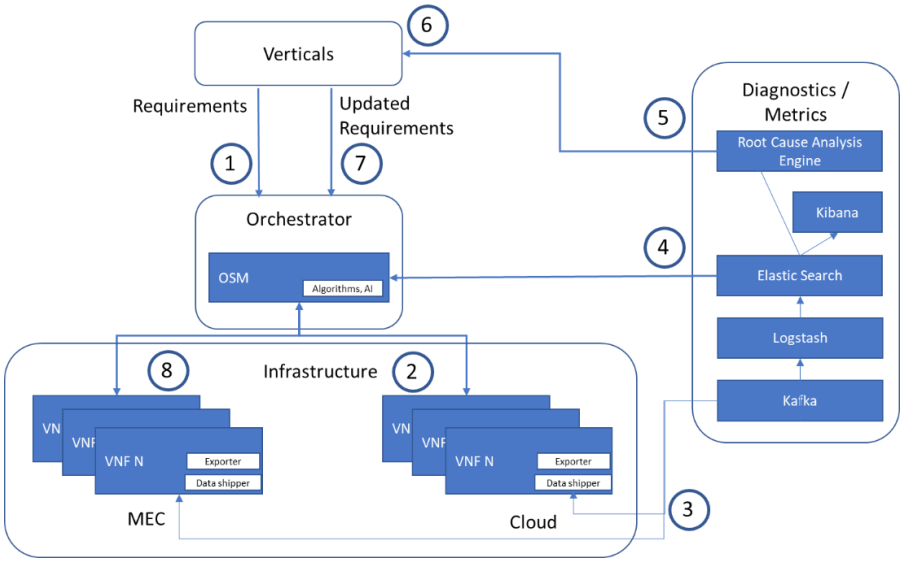


Figure 6.1.1. *Architecture of the AI-powered orchestration and of the associated diagnostics capabilities.*

The orchestrator's embedded AI will analyze the various data and historical information to find out if changes to the deployment of VNFs need to be done. If a metric does not meet the vertical requirement for a specific VNF, the orchestrator will make the necessary changes to fix it, either by relocation of the VNF, scaling or other appropriate actions.

A set of VNFs can be deployed at different MEC and Cloud infrastructures. In such a system, the end-users/IoT devices associated with a certain NS will have a random activity to transmit/receive data. Thus, the traffic demand of each NS is time-varying according to the number of devices that needs to be served at each time. VNFs typically do not require constant resources, but only when they serve traffic, and in fact consume resources commensurate to the traffic load. Hence, allocating a fixed number of resources often leads to either under- allocation or over-allocation of the system available resources. The ML algorithms (Support Vector Machines, Regression, Decision Tree, MLP Neural Network) are used when appropriate by the AI-enhanced MANO component to predict the number of resources required to successfully manage a given traffic load from a VNF instance. The model's prediction can be used by the VNF placement algorithm. Integrating this algorithm into a VNF placement algorithm for joint dynamic resource allocation and placement improves the VNF placement quality while avoiding the over-allocation and under-allocation. The contribution of AI-Enhanced MANO algorithmic approach can be summarized in the following steps:

- The learning model analyses and predicts the resource requirements of VNFs considering the traffic load at each time.
- The VNF placement algorithm considering the VNF resource requirements as predicted by the ML model, the infrastructure metrics/ available resources (e.g., latency, throughput, CPU, RAM, disk) and the service SLAs, minimizes the resulting end-to-end delay by deploying VNF instances close to their users.

The work of this subsection is complementary to that of previous subsections of chapter 6. Emphasis will be placed on the support of diverse AI algorithms (e.g., deciding on the placement), in order to enforce decisions and also to conduct monitoring and diagnostics regarding the efficiency of the decisions.

6.1.2 Mathematical problem formulation

Consider an infrastructure consisting of COTS servers and switches (which are for all intents and purposes are multihomed servers where only routing software can run) interconnected via network links. This infrastructure can be modelled as a directed graph with V_I the set of vertices and E_I the set of directed edges. Servers and switches in the infrastructure, jointly referred to as network nodes, are identified by a label $i \in V_I$ and the directed links by their start and end points $(i, i') \in E_I \subset V_I \times V_I$.

On this infrastructure NSs are hosted. NS m is modelled as an SFC, which is also modelled by a directed graph, with $V_{S,m}$ the set of its vertices and $E_{S,m}$ the set of its edges. Vertices of the SFC correspond to VNFs and edges correspond to the information flows that run between these VNFs. The VNFs of SFC m are identified by a label $k_m \in V_{S,m}$ and the information flow by their endpoints $(k_m, k'_m) \in E_{S,m} \subset V_{S,m} \times V_{S,m}$.

To activate NS m in the infrastructure its SFC needs to be embedded in the infrastructure. This means that the VNFs of SFC m need to be placed on network nodes and the information flows of SFC m to need to be routed over valid network paths. To describe this embedding two matrices of decision variables are defined as follows.

- A placement matrix x_m, where $x_{m,k_m,i} = 1$ if service function $k_m \in V_{S,m}$ is mapped to node $i \in V_I$ and $x_{m,k_m,i} = 0$ otherwise.
- Routing matrix y_m, where $y_{m,(k_m,k'_m),(i,i')} = 1$ if flow $(k_m, k'_m) \in E_S$ travels over link $(i,i') \in E_I$ and $y_{m,(k_m,k'_m),(i,i')} = 0$ otherwise.

Since each VNF of SFC m only needs to be placed once on a network node and an information flow needs to be routed over a valid path, the matrices x_m and y_m need to obey the following constraints:

$$\sum_{i \in V_I} x_{m,k_m,i} = 1 \quad \forall k_m \in V_{S,m}$$

$$x_{m,k_m,i} + \sum_{(i' | (i',i) \in E_I)} y_{m,(k_m,k'_m),(i',i)} = x^m_{k',i} + \sum_{(i' | (i.i') \in E_I)} y_{m,(k_m,k'_m),(i,i')} \leq 1 \quad \forall\, i \in V_I, (k_m, k'_m) \in E_{S,m}$$

The latter constraint specifies that in every network node i flow (k_m, k'_m) (associated to SFC m) can be either starting or entering in that node i, in which case it also needs to be terminated in or leaving that node, and this at most once. Notice that this is a necessary, but not sufficient condition for flow (k_m, k'_m) to be mapped on a valid path (as there still may be isolated loops).

At any moment in time the infrastructure hosts several NSs. A request for a new NS arrives from time to time and from time to time previously accepted service requests end. Let M be the set of the currently accepted NSs. The remaining compute (or memory) capacity C_i on the network nodes and the remaining transport capacity $R_{(i,i')}$ on the links is respectively:

$$C_i = C_{max,i} - \sum_{m \in M} \sum_{k_m \in V_{S,m}} w_{m,k_m} x_{m,k_m,i} \quad \forall i \in V_I$$

$$R_{(i,i')} = R_{max,(i,i')} - \sum_{m \in M} \sum_{(k_m,k'_m) \in V_{S,m}} f_{m,(k_m,k'_m)} y_{m,(k_m,k'_m),(i',i)} \quad \forall\, (i,i') \in E_I$$

where

- w_{m,k_m} is the compute (or memory) capacity required by the k_m-th VNF associated to SFC m;
- $f_{m,(k_m,k'_m)}$ is the bit rate required by flow (k_m, k'_m) associated to SFC m;
- $C_{max,i}$ the maximum compute (or memory) capacity of the i-the network node; and
- $R_{max,(i,i')}$ the maximum capacity of link (i,i');

all of which may fluctuate over time, the former two, because the demands of the NS m change over time and the latter two, because there may be other (i.e., $\notin M$) possibly higher priority processes or flows running on the infrastructure that consume capacity of the nodes or links that cannot be used by the set of active NSs ($m \in M$).

When a new NS request, say m' arrives, its associated SFC needs to be embedded, i.e., the decision matrices $x_{m'}$ and $y_{m'}$ need to be determined such that these remaining capacities are not overconsumed:

$$\sum_{k_{m'} \in V_{S,m'}} w_{m',k_{m'}} x_{m',k_{m'},i} \leq C_i \quad \forall i \in V_I$$

$$\sum_{(k_{m'},k'_{m'}) \in V_{S,m}} f_{m',(k_{m'},k'_{m'})} y_{m',(k_{m'},k'_{m'}),(i',i)} \leq R_{(i,i')} \quad \forall\, (i,i') \in E_I$$

Often there are other constraints involved as well, e.g., energy constraints (see Section 3.1), delay or reliability constraints.

Traditionally the problem above (i.e., determining $x_{m'}$ and $y_{m'}$ under the capacity constraints) is formulated as a constraint optimization problem, where the objective function reflects a cost associated to the embedding. This formulation assumes that the compute (or memory) capacity of all VNFs and the bit rate requirements of all flows associated to all active SFCs ($m \in M$) and the new (m') to be embedded SFC are known over their entire lifetime. Moreover, it is assumed that it is known when the currently active NSs will terminate. It is unrealistic to assume that all this information is available. That is the reason that ML (machine learning) techniques can be useful.

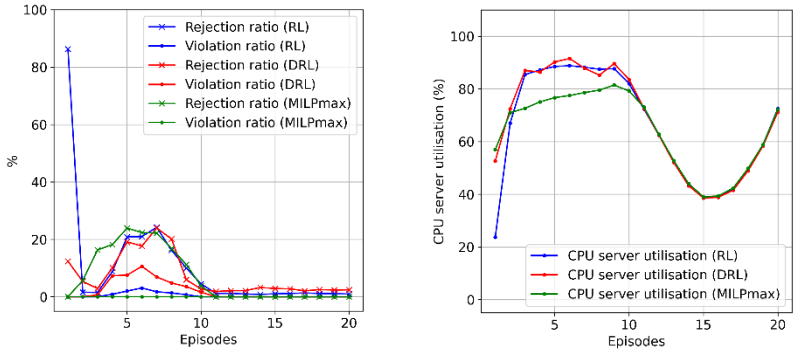
6.1.3 Reinforcement learning

To formulate the problem above as a RL instance, the state space, action space, rewards and transition laws need to be defined. The latter are often not explicitly needed and depend on the process with which new requests for NSs arrive, and on how they end. The state space encompasses all the information that is needed to fully specify the (capacity) constraints (not only at one specific instant, but over a sufficiently long interval). As we argued above, it is not realistic to assume that all this information is available. Therefore, the problem at hand is a Partially Observable Markov Decision Process (POMDP), so that, rather than the state space, we need to describe the observation space. The observation space consists of $(C_i, R_{(i,i')})$ observed at the instant that the new request arrives and an indication of how it will evolve over the coming time (and possibly an indication of the nature of the NS to be embedded if there are different types of slices). The action space is chosen such that it allows to determine $x_{m'}$ and $y_{m'}$. The set of all possible $x_{m'}$ and $y_{m'}$ is much too large to be used as action space directly. Therefore, the action space is a set of heuristics (e.g., rejection, collocated placement, random placement followed by Multi-Commodity Flow (MCF) routing, etc.) that allow to determine $x_{m'}$ and $y_{m'}$ very fast. With each heuristic a benefit is associated, e.g., low reward for rejection, medium reward for collocated placement, high reward for random placement followed by MCF routing. Subsequently, a penalty is deducted each time infrastructure resources are overconsumed. The reward associated with an embedding is the benefit minus these penalties.

6.1.4 Initial results

Figure **6.1.2** shows a typical example where the action space has two actions: accept and reject. This figure compares the rejection and violation ratio of three approaches:

- A traditional Mixed Integer Linear Programming (MILP) solution
- A traditional RL solution where the value that can be reached from a state is maintained in a table
- A Deep Reinforcement Learning (DRL) solution where the value that can be reached from a state is maintained in a Neural Network (NN), under a time-varying traffic demand (which follows a sinusoidal law). The figure shows that both RL solutions have about the same performance and outperform the traditional MILP solution. In particular, the rejection ratio (utilization) is higher (lower) for MILP, during a period of overload.



(a) *(b)*

Figure 6.1.2. *(a) Rejection and violation rate and (b) utility for three embedding approaches.*

More results are provided and discussed in more detail in [43].

6.1.5 Future work

As future work for this placement and routing topic, we will continue to explore other appropriate learning-based techniques for allocating resources to SFCs. In this context, an appropriate learning solution is one that (i) achieves a favorable trade-off between the identified pros and cons of existing methods, and (ii) is aligned with the objectives of the DAEMON research project. That is, we envisage the development of an explainable, reliable, and highly adaptable learning-based resource allocation scheme, which obtains close to optimal solutions within a few milliseconds.

One of the possible learning algorithms that we deem worth exploring are contextual multi-armed bandit algorithms. This intuition has its grounds on the following observations:

- Contextual multi-armed bandit algorithms are considered cutting-edge with respect to several decision-making areas, including recommendation systems.

- We can draw a direct analogy between the content recommendation problem space and the SFC-E problem space. Specifically, contextual multi-armed bandit algorithms can be leveraged to associate SFCs with embedding methods or pre-computed embedding solutions, in an analogy to users being associated with content.
- Recommendation engines are highly dynamic systems, e.g., users and content alter at a rapid pace. The successful application of contextual multi-armed bandit algorithms in this domain thus hints their potential to adjust quickly in highly variable environments, such as MANO systems.
- Contextual multi-armed bandit algorithms can deal with large and variable action spaces without accommodating any NN architecture. This adds to their explainability capacity, a feature missing from many modern NN-based learning techniques, such as deep Q-learning.
- To the best of our knowledge, contextual multi-armed bandit algorithms have not been thoroughly studied within the SFC-E context.

These render contextual multi-armed bandit algorithms an appealing and potentially viable approach to the problem in question.

6.2 Auto scaling

In this subsection, we consider the scenario in which SFCs (i.e., their associated VNFs are placed on COTS servers and their associated flows are routed) need additional or less resources (because they are consumed more or less than anticipated at when they were embedded). We consider two use cases: scaling of a single VNF and the coordinated scaling caches.

6.2.1 Auto scaling VNFs

Let us consider a network application that 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. Figure **6.2.1** gives an overview of the system previously described.

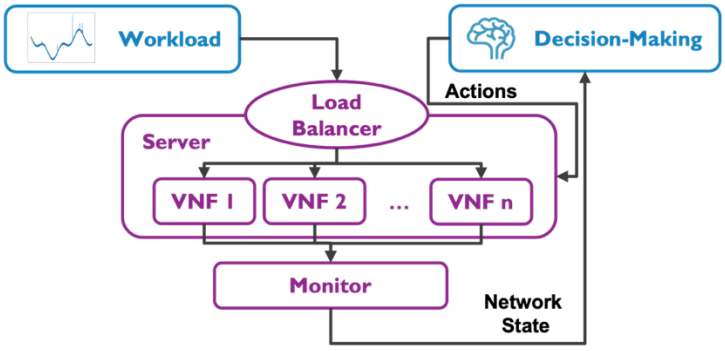


Figure 6.2.1. *System model.*

The auto-scaling problem follows a Monitor, Analyze, Plan and Execute (MAPE) loop. During the monitoring phase, the system monitor gathers and sends information about the system and the application state. This state is normally composed of infrastructure metrics (e.g., CPU usage and memory) and performance metrics experienced by the users (e.g., service latency, QoS). By using this information, the decision-making algorithm can estimate the future state (Analysis). This future state includes future resource utilization or expected workload. Then, the decision-making algorithm builds a function that maps the performance metrics with the scaling decisions (Planning). Finally, the system executes the actions planned by the decision-making algorithm.

As stated in the state of the art (see Appendix 8.4.3), several techniques have been proposed as decision-making algorithms, supervised and RL among those. Nonetheless, advanced predicting modules and modern RL approaches are nowadays based on DNNs, which require powerful computing processing. These resource-hungry DNNs might not be suitable for resource-constrained environments (e.g., Multi-access Edge Computing (MEC) hosts). Moreover, RL techniques require the ability to explore. In this way, the reward function can be maximized by taking random actions, exploring other areas in the solution space. However, this exploratory phase can lead to unstable agent behavior resulting in expensive networking errors.

In this regard, the selection of the applicable decision-making 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 amount of marginal penalty among agreed objectives, e.g., high penalty even when slightly violating the maximum service latency; therefore, the auto-scaler agent may have a higher chance to disregard the 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 ML-based auto-scaler is ruled out on a lower-end hardware. Instead, auto-scalers based on fixed rules and even control theory are computationally unexpensive algorithms that can provide good-quality solutions. Nonetheless, control theory is not designed with learning capabilities. Good values for its controlling parameters are often determined based on brute force or on linearizing the system to control.

6.2.2 Auto scaling Virtualized RAN (vRAN) caches

In the fast-emerging virtualized RANs, it is possible to rescale the network resources dynamically so as to adapt to the varying needs of users on the one hand and reduce the operating expenditures when such an opportunity arises, on the other hand. This approach is particularly useful for services that involve in-network storage of content or libraries or code. In those cases, indeed, one can rescale the cache dimension, increasing their capacity when there is need to store more files and reducing them when their cost (e.g., due to electricity prices) is high and/or the user needs are reduced. However, this dynamic scaling of vRAN resources brings new challenges. Firstly, typically the network needs to make these decisions without having access to future requests of the users or network conditions; and secondly, in order to reap all the benefits of this versatility, these decisions need to be made along with other network operation decisions such as routing, power control, or user-base station associations. This latter aspect adds an additional layer of complexity in the problem formulation and solution approach.

In the context of this project, we have developed a rigorous methodology for optimizing the performance of vRANs by jointly deciding: content caching at edge caches; user association to BSs; and cache scaling while adhering to a long-term monetary budget that is spent when leasing cache capacity. Our approach builds on Lyapunov-based stochastic optimization techniques [44], which we extend here to account for the discrete caching variables. The high-level system architecture is presented in Figure **6.2.2**, where we see a versatile RAN, with edge caches installed at the BSs. The main system parameters and key variables are presented in a succinct way within the figure and summarized also in Table 5.

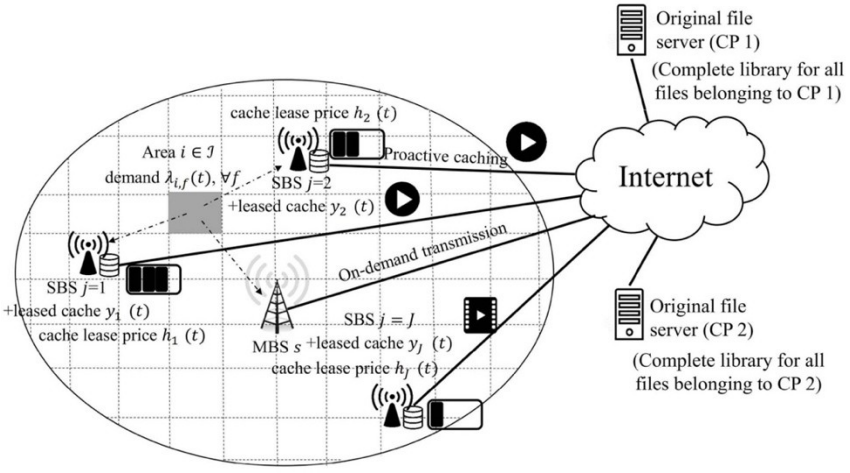


Figure 6.2.2: *Architecture and Main Concepts for Resource Rescaling in Virtualized RANs.*

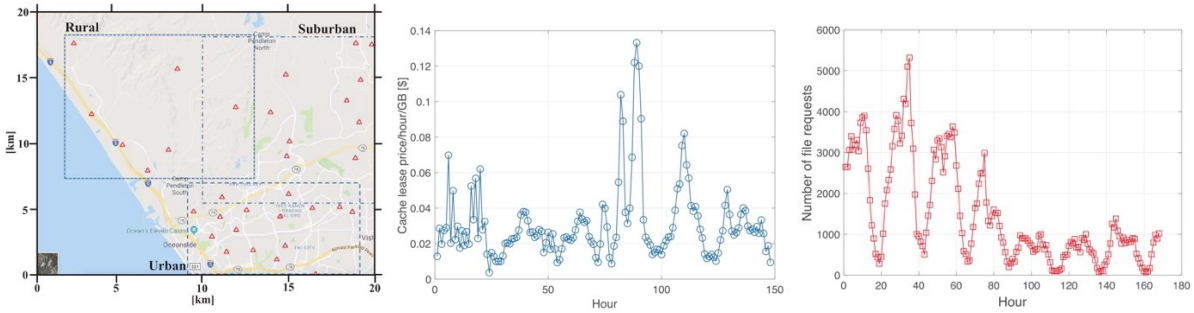
Table 5. *Main Parameters and Decision Variables.*

Notation	Definition	Notation	Definition
$i \in \mathcal{I}$	Subarea index	$h_j(t)$	Price to lease cache storage per unit bit for j and t
$i \in \mathcal{J}$	Small base station (SBS) index	$d_{ij}(t)$	Average delay for serving subarea i by SBS j during t
s	macro base station (MBS) index	$d_{is}(t)$	Average delay for serving subarea i by the remote server during t
$f \in \mathcal{F}$	File index	$x_{ij,f}(t)$	Association probability for i, j, f during t
B_{ave}	Average budget constrain	$y_j(t)$	Leased cache space at SBS f during t
$\lambda_{i,f}(t)$	Demand profile for I, f and t	$z_{j,f}(t)$	File caching indicator for j, f and t
t	Hour index (time slot)	$\gamma_i(t)$	Auxiliary variable of subarea i at time slot t
b	Size of a video file		

In detail, the goal of the network is to reduce as much as possible the latency for the delivery of the content files to the different mobile users; while not exceeding a predetermined average budget that it wishes to spend for deploying the caches. These costs represent either the leasing costs when the caches are owned by third parties; or the operating costs involving electricity and other similar OPEX factors. The system operation is considered time-slotted, where in the beginning of each slot the network decides the association, the leased cache space, and the file caching policy; then observes the performance of this specific configuration which is affected by the user requests and the network conditions (which determine the link quality for each user-BS pair), and the process repeats in the next slot. The network keeps track of these effects by means of (virtual) queues which monitor the "backlog" of each relaxed constraint (the monetary budget constraint, in particular) and which need to be asymptotically stable. The implementation details of the proposed algorithm can be found in [45].

The performance of the policy has been evaluated in a series of experiments with real data traces and has been proven quite satisfactory, outperforming the state of the art. In specific, we have used a dataset with real deployments of BSs in a rural and more urban setting, as shown in Figure **6.2.3** (a); a representative trace of cache leasing costs that is based on a pattern of electricity prices, Figure **6.2.3** (b); and a pattern of requests over time that is drawn from actual user request traces, Figure **6.2.3** (c).

The proposed model advances the state-of-the-art, see detailed review in Appendix, by focusing on a new and important problem, that of vRAN rescaling, where a joint optimization approach is taken across network and caching decisions. As a future direction, we plan to drop the stationarity assumption that is imposed both to the user requests and to the network state evolution and develop a new suite of online vRAN configuration algorithms that are robust to any type of demand patterns.



(a) *(b)* *(c)*

Figure 6.2.3. *(a) Rural and Suburban Deployment of BSs used in evaluation of vRAN rescaling Algorithm; (b) Cache leasing cost pattern; (c) Number of requests per hour.*

7 Conclusion and Outlook

In this deliverable, we have presented the initial design of intelligent orchestration and management mechanisms based on the architecture defined in DAEMON. Moreover, we identified four goals together with their interactions for the NI-based orchestration solution, and designed a general dataflow modeling method for decomposing the NI-based solutions into graphs for further NI orchestration purposes. This has led us to design 13 innovative NI-based solutions and to develop a three-stage plan for future refinement in forthcoming WP4 deliverables during the lifecycle of DAEMON. This document reports the initial design of these 13 solutions, presenting the challenges they raise, their high-level design objectives, their problem-solving approaches over the respective framework, and their preliminary performance evaluation and comparisons with existing (non-)NI-based solutions.

To exploit the full potential of the proposed NI-based solutions towards the milestone MS2 (Month13), where the initial design and validation campaign of NI functionalities is planned to be released, a joint effort is expected cutting through WP3, WP4, and WP5. Specifically, a selected subset of these 13 NI-based solutions will be presented to evaluate the potential deployment on a simulation/emulation platform or testbed in WP5 to show a thorough performance benchmark or even real-environment deployment. Similarly, the dataset provided in task T5.3 of WP5 can be treated as inputs to offline verify the functionality of NI-based solutions. In addition, the conclusion from studying these 13 solutions for management and orchestration will be fed back and investigated in more detail to provide the necessary information for the NI orchestration layer in WP2. Such information revealed to NI orchestration layer is necessary to make the deployment of each NI-based solution tailored to the needs of different use cases.

In a larger timescale, the initial design presented in this document will be further refined and extended in the next two WP4 deliverables, i.e., D4.2 (Month 23) and D4.3 (Month 33). Specifically, such refinements will not only provide a more detailed performance analysis of enhanced algorithms, but will also refresh the initial design based on the NI-based toolset and framework provided by WP2. Based on these refined solutions, we expect to set-up and disseminate a range of representative experimental proof-of-concepts, following the plan set at milestone MS6 (Month 30), to reflect the four identified goals defined in this document and justify the related DAEMON KPIs related to intelligent network orchestration and management. Ultimately, all NI-based solutions need to exploit the considered DAEMON framework, utilize the provided toolset to realize their objectives, and potentially explore the boundary of applying customized and adaptable AI to real-world NI problems.

8 Appendix A: State of the art

8.1 Energy-aware VNF orchestration

8.1.1 Energy-driven vRAN orchestration

Resource Orchestration in Networks. Existing works can be classified into: *(i)* studies that use models which relate control variables to performance; *(ii)* model-free approaches that rely on offline training data; and *(iii)* online learning techniques. Interesting examples in *(i)* include [46] which performs rate control to maximize throughput; [47] that selects also the MCS and airtime; and [48] that adapts to traffic. Nonetheless, such models are often unavailable and platform-dependent. On the other hand, model-free approaches employ machine learning to approximate the performance functions [49] and are used in slicing [20], throughput forecasting [50], etc. Their efficacy is remarkable as long as there are enough training data. Otherwise, we need to employ online learning. For instance, OCO is used for cloud and IoT resource orchestration, but requires convex functions; a condition not satisfied here. Another approach is RL, used in spectrum management [51], network diagnostics [52], and SDN control [53], among others. However, RL suffers from the curse of dimensionality, and lacks convergence guarantees.

Experimental Studies of vBS. The early work of [54] studied the cost savings when pooling the processing operations of BSs, and [55] proposed a vRAN architecture that reduces processing load by 30%. Other studies considered the effect of MCS, bandwidth, and SNR on Baseband Unit (BBU) computing load [56]. Using an OAI simulator, [57] models the processing time for different configurations and presented measurements with srsLTE for the impact of traffic. Our analysis builds on these important works and further measures the impact of new context parameters and radio schedulers on throughput, the coupling of uplink and downlink, and the vBS power consumption.

Analyses of network power consumption. Studies of power consumption in legacy BSs focus on the effect of power amplifier, RF output, and baseband processing. The *EARTH* model [16] relates the RF output power to supplied power; and [58] studies the effect of bandwidth. The works [59] [60] proposed models for macro and micro BSs, and [61] studied how the packet length affects the CPU power. Models accounting for various BS components are presented in [62]. Alas, for small vBSs other configuration parameters are equally important. Previous works focusing on vBS include [63] which considered the effect of CPU cores/speed, assuming a linear relation of traffic with computing load. This assumption is not always valid as our measurements and previous studies showed [56]. Importantly, the impact of hardware/software platform on these metrics cannot be captured in predetermined models. We overcome this obstacle by building models on-the-fly using the observed data.

8.2 Capacity forecasting

8.2.1 Unified state of the art on traffic and capacity forecasting

There exists a vast literature about prediction in networks [64], where the vast majority of works focus on foretelling future demands expressed in terms of plain traffic volumes. A variety of approaches have been proposed to solve the problem of mobile traffic forecasting, spanning exponential smoothing [65], autoregressive modelling [66], [67], [68], information theory [69], or Markovian processes [70].

In recent years, deep learning models have rapidly established themselves as the conventional tool for forecasting mobile network traffic. A plethora of diverse DNN architectures have been proposed to date, which leverage, among others, LSTM [71], [72], [73], Stacked AutoEncoder (SAE) [71], and MultiLayer Perceptron (MLP) [74] layers. Convolutional layers have been also extensively tested, in their vanilla [72], [73], three-dimensional [74], or graph [75] versions. These solutions have been used to predict traffic in different settings, including over short [71] and long [74] time horizons, or on aggregates [71], [74] and on a per-application basis [72]. For a comprehensive review, we refer the reader to surveys on applications of deep learning in networking, for forecasting and beyond [76]. We highlight, however, that comparative evaluations carried in the studies above have repeatedly proved that DNN models improve the quality of the prediction with respect to other approaches in general, and to statistical models in particular. This conclusion is also confirmed by dedicated performance analyses juxtaposing heterogeneous models [77], [78].

Several studies have also proposed predictors tailored to specific tasks in anticipatory networking, rather than generic traffic forecasting. These works have targeted forecasting for resource allocation [20], reconfiguration [21] and overbooking [79] in sliced networks, reservation of Physical Resource Blocks (PRBs) at the radio access [80], or assignment of baseband processing resources in Cloud Radio Access Network (C-RAN) [81]. Predictors have been also devised for physical layer indicators, such as bandwidth [82], transmission inactivity periods [83], or signal strength [84]. All such efforts are very recent, and exclusively rely on DNN architectures. Among others, they employ MLP [21], 3D convolutional [20], or regular [80] and multivariate [81] LSTM layers.

8.2.2 Prediction-assisted network scheduling for data analytics

Existing network control solutions fail to take into consideration the issues that arise when the network needs to offer edge analytic services. In particular, many approaches schedule the network resources using static optimization formulations and pre-determined models about the system operation [85]. However, the user requests and system states may vary abruptly, while the service performance depends on the system configuration and input data [86]. This also renders ineffective stochastic optimization approaches [87] that have been successful for stationary network scheduling problems [88]. Besides, most studies [89] do not account for the operation criteria of edge analytics, nor do they offer performance guarantees [90]. This creates a clear gap in the literature of network management that we will tackle in this task.

From an optimization perspective, the idea of using predictions has been explored in the recent years and is gaining increasing attention. For instance, [91] uses accurate predictions for the next cost function to improve the optimality gap, or regret, reduction rate down to logT, and [92] improved this result by allowing some predictions to be inaccurate. Alternatively, [93] follow a different approach by adapting to predictions' quality. For network management problems, in particular, predictions were included in stochastic optimization algorithms [94]which assume the requests and system perturbations are *stationary*; and in online learning algorithms [95] which *do not adapt* to the predictions' quality nor consider budget or other time-varying constraints. Hence, we plan to advance the state-of-the-art by proposing new algorithms with predictions that will address the open issues of these prior solutions.

8.3 Automated anomaly response

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

Our activities in DAEMON relate to network resilience [96], i.e., the property of the network to sustain its normal operation and desired performance, when facing several hard-to-predict situations. We tackle network residence in the context of global operations in the cellular domain.

One of the first aspects we consider is that of anomaly detection in a large telco carrier, which has long been a question of great interest in a wide range of domains including network systems [97] [98]. We are currently working to progress beyond the current state of the art, by studying machine learning and deep learning tools to implement anomaly detection functionalities that (i) include root cause analysis and response measures, (ii) are be trained with very large-scale measurement data, and (iii) will be prototyped in near-production systems; all these are original contributions with respect the current state of the art. The attribution of artificial intelligence and other machine learning models with resilience in the specific networking context requires novel technologies that address fundamental questions such as (i) whether these modules are operating based on valid, non-poisoned, data [99], and (ii) if the NI algorithms are producing proper, un-biased outputs, with explainable behavior [100], [101], [102].

The specific domain we investigate is that of an IPX Provider, which enables the interconnection of MNOs world-wide, and supports the data roaming function. Roaming has become significantly more common in Europe after the "Roam like Home" initiative came into effect and has spread to the other regions with different services (e.g., Google Fi). However, few studies have been conducted on roaming, possibly because its complex ecosystem and many involved parties bring about high costs and efforts for cooperation. Vallina et al. [103] studied national roaming between MNOs in France, and Michelinakis et al. [104] focused on international roaming between two operators in Europe. More recently, Mandalari et al. [105] covered international roaming more extensively, diving into the traffic among 16 MNOs in 6 different countries. The study shows that the technical solution for roaming that MNOs largely prefer is home-routed roaming (HR), which brings performance penalties on the roaming user, who experiences increased delay and appears to the public Internet as being connected in the home country. Despite these challenges, the growing demand for global mobile broadband access and a shift to all-IP-based services (from broadband last mile to VoIP) have brought new impetus to the old idea of the IPX, first proposed by the GSMA in 2007 to replace the traditional, bilateral-agreement model for international roaming [106]. Despite the continuous technical development by IPX-Ps and the related parties [107], there has been few academic works on the topic.

8.4 Self-leaning MANO

8.4.1 AI-enhanced MANO in B5G/6G

Previous work has been done in the context of the 5G EVE and 5G-Tours. Initial algorithms were designed. The work in 5G EVE addressed the issue of orchestrating 5G testbeds / network segments. 5G-Tours develops basic algorithms for managing computing resources, so as to prioritize the use of edge resources by "more important" services. Problem instances related to security and transportation (or entertainment) services are demonstrated, e.g., when some important security service needs to be supported, the orchestrator moves the lower priority services, e.g., transportation (or entertainment)

related, are moved to other resources. Work in Daemon will leverage and enhance the fundamental work, to encompass further applications and network technologies.

8.4.2 Placement and routing algorithms

The static version of the SFC embedding problem, where the required resources do not change over time, has been widely studied [108], [109]. The formulation of the problem varies, depending on the requirements of the NS, the application environment, and the level of detail of the monitoring information. Often the problem is formulated as a MILP, but sometimes non-linearities can be introduced depending on the NS intent (e.g., availability [110]), the non-linear communication network performance (e.g., functional split for RANs [111]), and the access technology (e.g., mmwave or Ethernet [112])

Overall, SFC embedding has been studied for various:

- Operational objectives: such as QoS [113] [114], profit maximization [115] [116], fault tolerance [117] [118] [119], load balancing [120] [121], energy efficiency [122], trust/security [123][], service mobility [124];
- Application domains: related to (i) different technology domains such as RANs [125], core networks [126] [127] [128] [129] and/or (ii) administrative domains such as single [130] [131] and multi-domain approaches [132].

SFC resource allocation problems have been formulated using RL in the past. In [133] the authors model requests as directed acyclic graphs and each request is broken into phases determined by the directed edges of the directed acyclic graph. The request is accepted if all phases succeed. The state models which node maps to which resource, and the actions represent moving nodes across resources. The system then learns optimal policies based on this Markov Decision Process (MDP).

In [134] each VNF is mapped to multiple nodes, and the state represents the VNFs mapped to each physical node. The action represents which VNF maps to which physical node. Higher reward is obtained by mapping VNFs to nodes with adequate capacity, rather than mapping to nodes that need to be scaled before-hand. The authors in [135] augment this model by classifying nodes as lightly loaded or heavily loaded. They also extend to the case where a SFC is mapped to a network of nodes. In [136] a strategy to map one VNF to a set of possible nodes is described. One of the algorithms in that paper trains a random NN to determine the best policy by giving a reward to a good decision, which is a form of direct policy learning. The authors in [137] formulate the network slicing resource allocation problem as a MDP and solve it with a policy gradient method. They model a network slice as a set of compute and transport resources without additional structure.

Up to now the system state and SFC demands were assumed to be static (or worst case was assumed). However, in reality they exhibit unpredictable variations due to stochastically arriving requests with different QoS requirements. Therefore, an adaptive online SFC deployment approach is needed to handle the real-time network variations and various service requests. In [138] the authors introduce a MDP model to capture the dynamic network state transitions and propose an adaptive, online, DRL approach to automatically deploy SFCs for requests with different QoS requirements.

All the aforementioned approaches described above embed VNFs one by one, while in DAEMON we aim to embed the whole SFC in a single step.

8.4.3 Scaling VNFs

Once the SFCs are deployed they consume resources on the infrastructure. As explained above, the SFC resource demand fluctuates over time. Decisions that were taken at the time the SFC was embedded may not be valid throughout the lifetime of the SFC. Therefore, from time-to-time scaling decision may have to be taking, migrating VNFs (and their associated flows) to other network nodes or allocating more resources to a certain VNF. These decisions can either be based on classical control algorithms [139], on predictions of how the resource demand will evolve in future or on RL [140]. In the following, we provide detailed review on different approaches/algorithms that can be applied in scaling decision-making.

Resource scaling can be classified as horizontal or vertical. In horizontal scaling, new VNF instances are added or removed as needed. In vertical scaling, the size of the VNF (e.g., its assigned computing and storage resources) is changed accordingly. Horizontal scaling can be exploited by distributed applications where the workload is shared among different instances of the same application. On the contrary, vertical scaling can be exploited by multi-threaded applications. While vertical scaling is limited to the capacity of a single server, horizontal scaling can leverage the virtual computing capacity in cloud environments, making it easier to deploy several instances of the same application. Additionally, horizontal scaling allows a more granular control of VNFs in multi-tier applications where each tier can be scaled independently of each other, e.g., three-tier architecture over presentation, logic, and data layers. Moreover, depending on the business model, acquiring more general-purpose hardware (i.e., adding more instances) is preferable to acquiring more powerful hardware due to its cost. Finally, as most

common OSs do not allow to change different properties of the VNF instance (e.g., assign more computing resources) on-the-fly, without rebooting, most cloud providers nowadays only offer horizontal scaling [141].

To go one further step, the scaling mode can be classified into proactive or reactive. Proactive scaling requires the ability to infer the upcoming workload so that the resources are scheduled beforehand, while in reactive scaling, the resources are changed in response to the perceived variations of a performance metric. It is worth noting that all the aforementioned scaling strategies try to find a balance between resource cost and fulfilling a given SLO. A network operator can easily guarantee an SLO by over-dimensioning the number of VNFs to serve the workload, but at the expense of increasing the economic cost. At the same time, the economic cost decreases when under-dimensioning the number of VNFs, but seriously compromise the SLO fulfillment.

Different techniques have been explored over the course of a decade for auto-scaling. Most of them, employ threshold-based rules to determine the right amount of VNF instances for serving a demand [142] [143] [144]. The main challenge in this type of auto-scaling technique is to create the rules that are guiding the decision-making algorithm. Frequently, these rules are created by experts to react appropriately when experiencing performance degradation, e.g., high service latency. Nevertheless, such auto-scaling approach can lead to ping pong effect and will need extra tuning to make a balance between reactiveness and stability. The popularity of threshold-based auto-scalers reside in their simplicity and ease of deployment; however, these reactive scalers are highly sensitive to sudden workload changes [145].

Control theory has also been explored for auto-scaling [146] [147] [148]. In this case, the controller relies in modelling the system to determine its future resource needs. The goal of the controller is to maintain a controlled variable (e.g., service latency) close to a desired level or reference level by adjusting a manipulated variable (e.g., number of VNF instances). The controlled variable is typically measured by a monitor or sensor and is considered the output of the system, while the manipulated variable is the input of the system to control. By adding a feedback loop, the controllers observe the changes of the system in response to changes in the input variable.

Recently, ML strategies are being proposed for flexible resource scaling in NFV-based networks given their ability to learn from data and past experiences. Most of the proposed strategies focused on proactive scaling [149] [150] [151] [152]. There are two ways of achieving proactive scaling. In the first approach, the ML module tries to predict the expected workload in the next management period. In this case, the workload is modeled as a time-series, where the ML module tries to forecast the trend using historical data. However, predicting the workload can be an easy task only if the workload follows regular patterns (e.g., during daytime, peak hours in weekdays, etc.). However, the prediction accuracy might degrade under unseen workload patterns. In the second approach, the auto-scaling problem is modeled as a classification problem. The ML module assigns a label (increase or decrease the number of VNFs) to a given combination of some input features (e.g., workload, usage of computing resources). However, building a training dataset for auto-scaling as a classification problem, requires defining how many VNF instances are needed to serve a given traffic load, which is a non-trivial task and requires expert knowledge. Additionally, the traffic load traces seen in testing might differ from the traces seen in training; therefore, a huge amount of training data must be available and labeled. Although it has been shown that proactive scaling produces better results in terms of reducing the boot-up time of a new instance and yielding few SLO violations, reactive scaling is still the widely used by the cloud industry due to its easiness to deploy, its decent performance, and its low computational cost.

RL is also explored as a solution for scaling network resources [153] [154] [155]. RL is a type of online-learning approach, where the agent does not need a labeled dataset since it learns from interacting with an environment. To determine the adequate amount of VNF instances to fulfill a maximum latency SLO using RL, the problem needs to be formulated as a Markovian Decision Process (MDP). The MDP is a discrete time stochastic framework for modeling decision-making problems. This process is defined by a tuple $(\mathcal{S}, \mathcal{A}, p, r)$ where $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of actions, p is the transition probability between states s and s_0 after action a is taken, and r is the immediate reward obtained for performing that action. The policy, defined as π, is a mapping function from states to actions. The solution to an MDP is an optimal policy that maximizes the expected long-term reward (which is a discounted sum of immediate rewards). In RL, the optimal policy is found after many interactions of the agent with the environment (i.e., in steady state). Q-Learning is the most used algorithm to find this optimal policy by defining a Q-function for all state-action pairs to measure how good the policy is. Therefore, finding the optimal policy is reduced to finding the action that maximizes this Q-function. Note that the state-action pairs can be stored in a table (i.e., Q-table); however, its size will growth substantially in accordance with the number of states (e.g., a continuous state space), preventing its wide adoption. Instead, Deep Q-Network (DQN) [156], uses a NN as Q-function approximator to overcome the limitation of the Q-Table's size.

8.4.4 Caching in vRAN

The original femtocaching model [157] presumes static content popularity and proposes a one-off proactive caching policy. Some recent studies in proactive caching, including our prior work, dropped the assumption of static popularity. The authors in [158] exploited Lyapunov optimization for proactive content caching and delivery in cellular and device-to-device networks, aiming to minimize the time-average network cost. Similarly, the work in [159] addressed the issue of non-stationary content popularity using an online learning approach in the design of the routing and caching policies. Nevertheless, the above works overlooked the possibility of cache scaling, which is a central issue in self-learning MANO.

On the other hand, reactive caching policies, such as LRU (Least Recently Used) and LFU (Least Frequently Used) make dynamic caching decisions upon the arrival of each request. These policies have been originally designed for single (or, independent) caches, and have been later extended to caching networks. For instance, [160] proposed a q-LRU algorithm where the caching update happens depending on the cached status in multiple BSs. [161] addressed a stochastic cooperative caching in several BSs under assumption of coded and Time-To-Live (TTL) caching aiming at reducing content download time. Similar to our work, [162] addressed the dynamic cache resource scaling aiming to simultaneously minimize the storage and backhaul cost. However, unlike our work, they considered a reactive caching policy, namely TTL caching, single cache and non-fairness criteria. Moreover, [163] considered a utility-based objective function where the utility captures fairness between different files. They proposed utility-driven LRU and LFU algorithms aiming at maximizing the sum of utilities over all files without consideration of cache scaling. Besides, our utility function is designed to achieve user - not file - fairness

Past content caching works using machine learning, e.g., collaborative filtering, focused on accurate estimation of future content popularity [164] . Recently, several studies exploited (deep) RL to optimize the operation of caching networks. For example, [165] addressed content caching in broadcasting systems using DRL. Moreover, [166] used RL as a content caching solution in a unicast system where their storage price might change with time. In their previous work, [167]considered a hierarchical cloud-edge caching model where the cloud stores files according to global file popularity and the edge stores files according to local file popularity. They modeled the spatio-temporal popularity variations using a Markov chain model and solved them with RL. Finally, the work [168] proposed a mobile proactive caching scheme, using again RL, where the caches are deployed at the mobile users' equipment, not at edge servers as in our model.

The above works do not consider the envisioned elasticity in automatically rescaling RAN infrastructures in B5G and 6G systems; nor the economic issues arising by the operating expenditures (or, leasing costs) of such dynamic network infrastructures. The work conducted in the context of DAEMON WP4 fills this gap, in a satisfactory way, as it is manifested by a range of data-driven comparisons with carefully-selected benchmarks.

9 References

[1] I. Paez and L. Cominardi, "D2.1 Initial report on requirements analysis and state-of-the-art frameworks and toolsets," DAEMON, 2021.

[2] W. Johnston, J. Hanna and R. Millar, "Advances in dataflow programming languages," *ACM Computing Surveys,* vol. 36, no. 1, pp. 1-34.

[3] G. Baldoni, J. Loudet, L. Cominardi, A. Corsaro and Y. He, "Facilitating distributed data-flow programming with Eclipse Zenoh: the ERDOS case.," in *1st Workshop on Serverless mobile networking for 6G Communications* , Virtual (USA), 2021.

[4] A. Cañete, M. Amor and L. Fuentes, "Energy-efficient Deployment of IoT Applications in Edge-based Infrastructures: A Software Product Line Approach," *IEEE Internet of Things Journal (Early Access),* 2020.

[5] K. Kang, J. Lee, S. Kim, E. Shin and M. Huh, "Form: A feature-oriented reuse method with domain-specific reference architectures," *Annals of Software Engineering,* vol. 5, p. 143–168, 1998.

[6] Y. Mao, C. You, J. Zhang, K. Huang and K. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys Tutorials,* vol. 19, no. 4, pp. 2322–2358,, 2017.

[7] T. Dinh, Q. Tang, Q. La and Q. Quek, "Offloading in mobile edge computing: Task allocation and computational frequency scaling,," *IEEE Transactions on Communications,* vol. 65, no. 8, p. 3571–3584, 2017.

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

[9] W. Zhang, Y. Wen and D. Wu, in *IEEE INFOCOM*, 2013.

[10] S. Melendez and M. P. McGarry, in *14th IEEE Annual Consumer Communications Networking Conference (CCNC)*, 2017.

[11] A. Niewiadomski, J. Skaruz, W. Penczek, M. Szreter and M. Jarocki, "SMT versus genetic and OpenOpt algorithms: Concrete planning in the PlanICS framework," *Fundamenta Informaticae,* vol. 135, p. 451–466, 2014.

[12] N. Bjørner, A. D. Phan and L. Fleckenstein, "vz - an optimizing SMT solver," *Tools and Algorithms for the Construction and Analysis of Systems,* p. 194–199, 2015.

[13] W. Shi, J. Cao, Q. Zhang, Y. Y. Li and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal,* vol. 3, no. 5, p. 637–646, 2016.

[14] I. Gomez-Miguelez, A. Garcia-Saavedra, P. D. Sutton, P. Serrano, C. Cano and D. J. Leith, "srsLTE: an open-source platform for LTE evolution and experimentation," in *10th ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization*, 2016.

[15] A. Garcia-Saavedra and X. Xavier Costa-Perez, " O-RAN: Disrupting the Virtualized RAN Ecosystem.," *IEEE Communications Standards Magazine,* 2021.

[16] G. Auer, V. Giannini, C. Desset, I. Godor, P. Skillermark, M. Olsson, M. A. Imran, D. Sabella, M. J. Gonzalez, O. Blume and A. Fehske, "How Much Energy is Needed to Run a Wireless Network?," *IEEE Wireless Communications,* vol. 18, no. 5, p. 40–49, 2011.

[17] J. A. Ayala-Romero, A. Garcia-Saavedra, M. Gramaglia, X. Costa-Perez, A. Banchs and J. J. Alcaraz, "vrAIn: Deep Learning based Orchestration for Computing and Radio Resources in vRANs," *IEEE Transactions on Mobile Computing,* 2020.

[18] G. Garcia-Aviles, A. Garcia-Saavedra, M. Gramaglia, P. Serrano, X. Costa-Perez and A. Banchs, "Nuberu: Reliable RAN Virtualization in Shared Platfroms," in *27th Annual International Conference on (ACM MobiCom '21)*, 2021.

[19] I. Gomez-Miguelez, A. Garcia-Saavedra, P. D. D Sutton, P. Serrano, C. Cristina Cano and D. J. Leith, "srsLTE: an open-source platform for LTE evolution and experimentation," in *10th ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization*, 2016.

[20] D. Bega, M. Gramaglia, M. Fiore, A. Banchs and X. X. Costa-Perez, "DeepCog: Cognitive Network Management in Sliced 5G Networks with Deep Learning," in *IEEE INFOCOM*, Paris (France), 2019.

[21] D. Bega, M. Gramaglia, M. B. A. Fiore and X. X. Costa-Perez, "Aztec: Anticipatory capacity allocation for zero-touch network slicing," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, 2020.

[22] S. Makridakis, E. Spiliotis and V. Assimakopoulos, "The m4 competition: 100,000 time series and 61 forecasting methods , vol. 36, no. 1, pp. ,," *International Journal of Forecasting,* vol. 36, no. 1, p. 54 – 74, 2020.

[23] S. Smyl, "A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting," *International Journal of Forecasting,* vol. 36, no. 1, p. 75 – 85, 2020.

[24] R. Clemen, "Combining forecasts: A review and annotated bibliography," *International Journal of Forecasting,* vol. 5, no. 4, p. 559 – 583, 1989.

[25] O. Sagi and L. L. Rokach, "Ensemble learning: A survey," *WIREs Data Mining and Knowledge Discovery,* vol. 8, no. 4, p. e1249, 2018.

[26] J. Nightingale, P. Salva-Garcia, J. Calero and Q. Q. Wang, "5G- QOE: Qoe modelling for ultra-hd video streaming in 5g networks," *IEEE Transactions on Broadcasting,* vol. 64, no. 2, p. 621–634, 2018.

[27] 3GPP, "Architecture enhancements for 5G system (5GS) to support network data analytics services," ETSI, 2020.

[28] D. Chatzopoulos, C. Bermejo, Z. Huang and P. Hui, "Mobile Augmented Reality Survey: From Where We Are to Where We Go," *IEEE Access,* pp. vol. 5, pp. 6917–6950, 2017.

[29] Y. Wang, "Enabling Edge-Cloud Video Analytics for Robotics Applications," in *INFOCOM*, 2020.

[30] H. Zhang and et al., "Live Video Analytics at Scale with Approximation and Delay-Tolerance," in *NSDI*, 2017.

[31] S. Yi and et al., "LAVEA: Latency-aware Video Analytics on Edge Computing Platform," in *Symp. on Edge Computing*, 2017.

[32] G. Paschos and et al., "Wireless caching: Technical Misconceptions and Business Barriers," *IEEE Communications Magazine,* pp. vol. 54, no. 8, pp. 16–22, 2016.

[33] X. Chen and et al., "Efficient Multi-User Computation Offloading for Mobile-Edge Cloud Computing," *IEEE/ACM Trans. on Networking,* pp. vol. 24, no. 5, pp. 2795–2808, 2016.

[34] A. K. S. Anitha Ramchandran, "Chapter 11 - Unsupervised Anomaly Detection for High Dimensional Data—an Exploratory Analysis," in *Intelligent Data-Centric Systems, Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Apllications*, A. K. Sangaiah, M. Sheng and Z. Zhang, Eds., Academic Press, 2018, pp. 233-251.

[35] M. Goldstein and S. Uchida, "A Comparative Evaluation of Unsupervised Anomaly Detection Algorithms for Multivariate Data," *PLoS ONE,* vol. 11, no. 4, p. e0152173, 2016.

[36] K. G. Mehrotra, C. K. Mohan and H. Huang, "Anomaly Detection Principles and Algorithms," *Springer Nature,* 2017.

[37] 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.

[38] A. Lutu, B. Jun, A. Finamore, F. E. Bustamante and D. Perino, "Where things roam: Uncovering cellular IoT/M2M connectivity," in *ACM Internet Measurement Conference (IMC '20)*, NewYork (NY, USA), 2020.

[39] A. Lutu, B. Jun, F. E. Bustamante, D. Perino, M. Bagnulo and C. G. Bontje, "A first look at the IP exchange ecosystem," *ACM SIGCOMM Computer Communication Review,* vol. 50, no. 4, p. 25–34, 2020.

[40] A. Lutu, D. D. Perino, M. Bagnulo and F. Bustamante, "Insights from Operating an IP eXchage Provider," in *ACM SIGCOMM 2021 Conference (SIGCOMM '21)*, New York (NY, USA), 2021.

[41] ETSI, "Open Source MANO (OSM)".

[42] ETSI, "Network Functions Virtualisation (NFV); Management and Orchestration," ETSI, 2014.

[43] T. Wassing, D. De Vleeschauwer and C. Papagianni, "A Machine Learning Approach for Service Function Chain Embedding in Cloud Datacenter Networks," in *IEEE International Conference on Cloud Networking (Cloudnet21)*, Virtual Conference, 2021.

[44] L. Georgiadis, M. Neely and L. Tassiulas, "Resource Allocation and Cross-Layer Control in Wireless Networks," *Found. Trends Netw.,* p. 1(1), 2006.

[45] J. Kwak, G. Paschos and G. Iosifidis, "Elastic FemtoCaching: Scale, Cache, and Route," *IEEE Trans. Wirel. Commun,* pp. 20(7): 4174-4189, 2021.

[46] P. Rost, M. Maeder, M. C. Valenti and S. Talarico, "Computationally Aware Sum-Rate Optimal Schedulingfor Centralized Radio Access Networks," in *IEEE Global Communications Conference (GLOBECOM'15)*, 2015.

[47] D. Bega, A. Banchs, M. Gramaglia, X. Costa-Pérez and P. Rost, "CARES: Computation-aware Scheduling in VirtualizedRadio Access Networks," *IEEE Transactions on Wireless Communications,* vol. 17, no. 12, p. 7993–8006, 2018.

[48] K. Wang, X. Yu, W. Lin, Z. Deng and X. Liu, "Computing Aware Scheduling in Mobile Edge Computing System," *Springer Wireless Networks,* 2019.

[49] C. Zhang, P. Patras and H. Haddadi, "Deep Learning in Mobile and Wireless Networking: A Survey," *IEEE Communications Surveys & Tutorials,* vol. 21, no. 3, pp. 2224-2287, 2019.

[50] D. Raca, A. H. Zahran, C. J. Sreenan, R. K. Sinha, E. Halepovic, R. Jana and V. Gopalakrishnan, "On Leveraging Machine and Deep Learning for Throughput Prediction in Cellular Networks: Design, Performance, and Challenges," *IEEE Communications Magazine,* vol. 58, no. 3, pp. 11-17, 2020.

[51] N. Zhao, Y. Liang, D. Niyato, Y. Pei, M. W. M. and Y. Jiang, "Deep Reinforcement Learning for User Association and Resource Allocation in Heterogeneous Cellular Networks," *IEEE Trans.on Wireless Communications,* vol. 18, no. 11, pp. 5141-5152, 2019.

[52] F. Mismar, J. Choi and B. L. Evans, "A Framework for AutomatedCellular Network Tuning With Reinforcement Learning," *IEEE Trans-actions on Communications,* vol. 67, no. 10, 2019.

[53] Z. Zhang, L. Ma, K. Poularakis, K. K. Leung and L. Wu, "DQ scheduler: Deep reinforcement learning based controller synchronizationin distributed SDN," in *International Conference on Communications (ICC'2019)*, 2019.

[54] S. Bhaumik, S. P. Chandrabose, M. K. Jataprolu, G. Kumar, A. Muralidhar, P. Polakos, V. Srinivasan and W. Thomas, "CloudIQ: A Framework for Processing Base stations in a Data Center," in *18th annual international conference on Mobile computing and networking (MobiCom'12)*, New York (NY, USA), 2012.

[55] W. Wu, L. E. Li, A. Panda and S. Shenker, "PRAN: Programmable Radio Access Networks," in *13th ACM Workshop on Hot Topics in Networks (HotNets-XIII)*, New York (NY, USA), 2014.

[56] P. Rost, S. Talarico and M. C. Valenti, "The complexity–rate tradeoffof centralized radio access networks," *IEEE Transactions on Wireless Communications,* vol. 14, no. 11, 2015.

[57] N. Nikaein, "Processing Radio Access Network Functions in the Cloud: Critical Issues and Modeling," in *Workshop on Mobile Cloud Computing and Services*, 2015.

[58] H. Holtkamp, G. Auer, V. Giannini and H. Haas, "A parameterized base station power model," *IEEECommunications Letters,* vol. 17, no. 11, pp. 2033-2035, 2013.

[59] O. Arnold, F. Richter, G. Fettweis and O. Blume, "Power Consumption Modeling of Different Base Station Types in Heterogeneous Cellular Networks," in *IEEE Future Network and Mobile Summit*, 2010.

[60] M. Deruyck, W. Joseph and L. Martens, "Power Consumption Model for Macrocell and Microcell Base Stations," *Transactions on EmergingTelecommunications Technologies,* vol. 25, no. 3, pp. 320-333, 2014.

[61] B. H. Jung, H. Leem and D. K. Sung, "Modeling of Power Consumption for Macro-, Micro-, and RRH-based Base Station Architectures," in *79th Vehicular Technology Conference (VTC Spring)*, 2014.

[62] C. C. Desset, B. Debaillie, V. Giannini, A. Fehske, G. Auer, H. Holtkamp, W. Wajda, D. Sabella, F. Richter, M. Gonzalez, H. Klessig, I. Gódor, M. Olsson, I. M.A., A. Ambrosy and O. Blume, "Flexible

power modeling of LTE base stations," in *Wireless Communications and Networking Conference (WCNC'12)*, 2012.

[63] T. Zhao, J. Wu, S. Zhou and Z. Niu, "Energy-delay tradeoffs of virtual base stations with a computational-resource-aware energy consumption model," in *International Conference on Communication Systems (ICCS'14)*, 2014.

[64] M. Joshi and T. Hadi, "A review of network traffic analysis and prediction techniques," arXiv preprint, 2015.

[65] D. Tikunov and T. Nishimura, "Traffic prediction for mobile network using holt-winter's exponential smoothing," in *15th International Conference on Software, Telecommunications and Computer Networks*, 2007.

[66] F. Xu, Y. Lin, J. Huang, D. Wu, H. Shi, J. Song and Y. Li, "Big Data Driven Mobile Traffic Understanding and Forecasting: A Time Series Approach," *IEEE Transactions on Services Computing,* vol. 9, p. 796–805, 2016.

[67] S. Ntalampiras and M. Fiore, "Forecasting mobile service demands for anticipatory MEC," in *19th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*, 2018.

[68] M. Zhang, H. Fu, Y. Li and S. S. Chen, ""Understanding urban dynamics from massive mobile traffic data," *IEEE Transactions on Big Data,* vol. 5, no. 2, p. 266–278, 2019.

[69] R. Li, Z. . Zhao, X. Zhou, J. Palicot and H. H. Zhang, "The prediction analysis of cellular radio access network traffic: From entropy theory to networking practice," *IEEE Communications Magazine,* vol. 52, no. 6, p. 234–240, 2014.

[70] M. Shafiq, L. Ji, A. Liu and J. J. Wang, "Characterizing and modeling internet traffic dynamics of cellular devices," in *ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, 2011.

[71] J. Wang, J. Tang, Z. W. Y. Xu, G. Xue, X. Zhang and D. D. Yang, "Spatiotemporal modeling and prediction in cellular networks: A big data enabled deep learning approach," in *IEEE International Conference on Computer Communications (IEEE INFOCOM)*, 2017.

[72] C. Zhang, M. Fiore and P. P. Patras, "Multi-Service Mobile Traffic Forecasting via Convolutional Long Short-Term Memories," in *IEEE International Symposium on Measurements and Networking (IEEE M&N)*, Catania, Italy, 2019.

[73] C. Huang, C. Chiang and Q. Q. Li, "A study of deep learning networks on mobile traffic forecasting," in *28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, 2017.

[74] C. Zhang and P. Patras, "Long-term mobile traffic forecasting using deep spatio-temporal neural networks," in *Eighteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing (Mobihoc)*, 2018.

[75] X. Wang, Z. Zhou, F. Xiao, K. Xing, Z. Yang, Y. Liu and C. C. Peng, "Spatio-temporal analysis and prediction of cellular traffic in metropolis," *IEEE Transactions on Mobile Computing,* vol. 18, no. 9, p. 2190– 2202, 2019..

[76] B. Ma, W. Guo and J. J. Zhang, "A survey of online data-driven proactive 5g network optimisation using machine learning," *IEEE Access,* vol. 8, p. 35606–35637, 2020.

[77] J. Yin, W. Rao, M. Yuan, J. Zeng, K. Zhao, C. Zhang, J. Li and Q. Zhao, "Experimental study of multivariate time series forecasting models," in *28th ACM International Conference on Information and Knowledge Management (CIKM19)*, New York (US), 2019.

[78] S. Sone, J. Lehtomaki and Z. Z. Khan, "Wireless traffic usage forecasting using real enterprise network data: Analysis and methods," *IEEE Open Journal of the Communications Society,* vol. 1, pp. 777-797, 2020.

[79] J. Salvat, L. Zanzi, A. Garcia-Saavedra, V. Sciancalepore and X. X. Costa-Perez, "Overbooking Network Slices Through Yield-driven End-to-end Orchestration," in *14th International Conference on emerging Networking EXperiments and Technolologies (ACM CoNEXT18)*, 2018.

[80] C. Gutterman, E. Grinshpun, S. Sharma and G. Zussman, "RAN resource usage prediction for a 5G slice broker," in *20th ACM International Symposium on Mobile Ad Hoc Networking and Computing (Mobihoc19)*, (New York, NY, USA).

[81] L. Chen, T. Nguyen, D. Yang, M. Nogueira, C. Wang and D. Zhang, "Data-Driven C-RAN Optimization Exploiting Traffic and Mobility Dynamics of Mobile Users," *IEEE Transactions on Mobile Computing,* 2020.

[82] C. Yue, R. Jin, K. Suh, Y. Qin, B. Wang and W. Wei, "LinkForecast: Cellular link bandwidth prediction in LTE networks," *IEEE Transactions on Mobile Computing,* vol. 17, no. 7, p. 1582–1594.

[83] P. Brand, J. Falk, J. Ah Sue, J. Brendel, R. Hasholzner and J. J. Teich, "Adaptive predictive power management for mobile LTE devices," *IEEE Transactions on Mobile Computing,* pp. 1-18, 2020.

[84] A. Kulkarni, A. Seetharam, A. Ramesh and J. Herath, "DeepChannel: Wireless channel quality prediction using deep learning," .," *IEEE Transactions on Vehicular Technology,* vol. 69, no. 1, p. 443–456, 2020.

[85] T. Tan and C. Gao, "Fastva: Deep learning video analytics through edge processing and npu in mobile," in *INFOCOM*, 2020.

[86] A. Galanopoulos, J. Ayala, D. Leith and G. Iosifidis, "AutoML for Video Analytics with Edge Computing," in *INFOCOM*, 2021.

[87] C. Wang and et al., "Joint Configuration Adaptation and Bandwidth Allocation for Edge-based Real-time Video Analytics," in *INFOCOM*, 2020.

[88] M. Neely, Stochastic Network Optimization with Application to Communication and Queueing Systems. Synthesis Lectures on Communication Networks, Morgan and Claypool Publishers, 2010.

[89] J. Meng and et al., "Adaptive User-managed Service Placement for Mobile Edge Computing: An Online Learning Approach," in *INFOCOM*, 2019.

[90] X. Ran and et al., "DeepDecision: A Mobile Deep Learning Framework for Edge Video Analytics," in *INFOCOM*, 2018.

[91] O. Dekel, A. Flajolet, N. Haghtalab and P. Jaille, "Online Learning with a Hint," in *NIPS*, 2017.

[92] A. Bhaskara, A. Cutkosky and et al., "Online Learning with Imperfect Hints,," in *ICML*, 2020.

[93] S. Rakhlin and K. Sridharan, "Optimization, Learning, and Games with Predictable Sequences," in *NIPS*, 2013.

[94] K. Chen and L. Huang, "Timely-Throughput Optimal Scheduling With Prediction," *IEEE/ACM Tran. on Networking,* p. 26(6), 2018.

[95] Z. Zhous, X. X. Chen and e. al., "Predictive Online Server Provisioning for Cost-Efficient IoT Data Streaming Across Collaborative Edges," in *ACM Mobihoc*, 2019.

[96] P. Vlacheas, V. Stavroulaki, P. Demestichas, S. Cadzow, D. Ikonomou and S. Gorniak, "Towards end-to-end network resilience," *Int. J. Crit. Infrastruct. Prot.,* 2013.

[97] M. Ahmed, A. N. Mahmood and J. Hu, "A survey of network anomaly detection techniques," *J. Netw. Comput. Appl,* vol. 60, 2016.

[98] V. Chandola, A. Banerjee and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.,* vol. 41, 2009.

[99] M. Barreno, B. Nelson, A. D. Joseph and J. D. Tygar, "The security of machine learning," *Mach. Learn.,* vol. 81, 2010.

[100] Z. C. Lipton, "The mythos of model interpretability," *Queue,,* vol. 16, 2018.

[101] S. Raaijmakers, "Artificial Intelligence for Law Enforcement: Challenges and Opportunities," *IEEE Secur. Priv.,* vol. 17, 2019.

[102] J. Silberg and J. Manyika, "Notes from the AI frontier: Tackling bias in AI (and in humans)," *McKinsey Glob. Inst.,* 2019.

[103] N. Vallina-Rodriguez, S. Sundaresan, C. Kreibich, N. Weaver and V. Paxson, "Beyond the Radio: Illuminating the Higher Layers of Mobile Networks," in *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, 2015.

[104] F. Michclinakis, H. Doroud, A. Razaghpanah, A. Lutu, N. Vallina-Rodriguez, P. Gill and J. Widmer, "The Cloud that Runs the Mobile Internet: A Measurement Study of Mobile Cloud Services," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, 2018.

[105] A. M. Mandalari, A. Lutu, A. Custura, A. S. Khatouni, O. Alay, M. Bagnulo, V. Bajpai, A. Brunstrom, J. Ott, M. Mellia and G. Fairhurst, "Experience: Implications of Roaming in Europe," in *24th Annual International Conference on Mobile Computing and Networking*, 2019.

[106] GSM Association, "IR.34-Guidelines for IPX Provider networks, Version 9.1," December 2019. [Online]. Available: https://www.gsma.com/newsroom/wp-content/uploads/2013/05/IR.34-v9.1.pdf}. [Accessed 01 04 2021].

[107] R. Xu, H. J. Tang and A. Joseph, "Method and System For Hub Breakout Roaming". Patent US Patent US20140169286A1, 2014.

[108] J. Herrera and J. F. Botero, "Resource allocation in NFV: A comprehensive survey," *IEEE Transactions on Network and Service Management,* vol. 13, no. 3, p. 518–532, 2016.

[109] B. Yi, X. Wang, X., K. Li and M. Huang, "A comprehensive survey of network function virtualization,"," *Computer Networks,* vol. 133, p. 212–262, 2018.

[110] S. Herker, A. X., W. Kiess and A. Kirstadter, "Path protection withexplicit availability constraints for virtual network embedding," in *IEEE 24th Annual International Symposium on Personal, Indoor, andMobile Radio Communications (PIMRC)*, 2013.

[111] B. M. Khorsandi, D. Colle, W. Tavernier and C. Raffaelli, "Adaptive function chaining for efficient design of 5G xhaul," in *International IFIP Conference on Optical Network Design and Modeling*, 2019.

[112] N. Akhtar, I. Matta, A. Raza, L. Goratti, T. Braun and F. Esposito, "Virtual Function Placement and Traffic Steering over 5G Multi-Technology Networks," in *4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, 2018.

[113] M. Huang, W. Liang, Y. Ma and S. Guo, " Throughput maximization of delay-sensitive request admissions via virtualized network function placements and migrations," in *IEEE International Conference on Communications (ICC),*, 2018.

[114] D. Cho, J. Taheri, A. Zomaya and P. Bouvry, "Real-time virtualnetwork function (VNF) migration toward low network latency in cloud environments," in *IEEE 10th International Conference on Cloud Computing (CLOUD)*, 2017.

[115] W. Racheg, N. Ghrada and M. Zhani, "Profit-driven resource provisioning in NFV-based environments," in *IEEE International Conference on Communications (ICC)*, 2017.

[116] Y. Ma, W. Liang, Z. Xu and S. Guo, "Profit maximization for admitting requests with network function services in distributed clouds," *IEEE Transactions on Parallel and Distributed Systems,* 2018.

[117] S. Bijwe, F. Machida, S. Ishida and S. Koizumi, "End-to-end reliability assurance of service chain embedding for network function virtualization," in *IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, 2017.

[118] B. Yang, Z. Xu, W. Chai, W. Liang, D. Tuncer, A. Galis and G. G. Pavlou, "Algorithms for fault-tolerant placement of stateful virtualized network functions," in *IEEE International Conference on Communications (ICC)*, 2018.

[119] J. Khalid and A. Akella, "Correctness and performance for stateful chained network functions," in *16th USENIX Symposium on Networked Systems Design and Implementation*, Boston, 2019.

[120] F. Carpio, S. Dhahri and A. Jukan, "VNF placement with replication for load balancing in NFV networks," in *IEEE International Conference on Communications (ICC)*, 2017.

[121] X. X. Fei, F. Liu, H. Xu and H. Jin, "Towards load-balanced VNF assignment in geo-distributed NFV infrastructure," in *IEEE/ACM 25th International Symposium on Quality of Service (IWQoS)*, 2017.

[122] F. Eramo, M. Ammar and F. F. G. Lavacca, "Migration energy aware re-configurations of virtual network function instances in NFV architectures,," *IEEE Access,* vol. 5, p. 4927–4938, 2017.

[123] N. Torkzaban, C. Papagianni and J. S. Baras, "Trust-aware service chain embedding," in *6th International Conference on Software Defined Systems (SDS)*, 2019.

[124] C. Y.-T. and W. W. Liao, "Mobility-Aware Service Function Chaining in 5G Wireless Networks with Mobile Edge Computing," in *IEEE International Conference on Communications (ICC),*, 2019.

[125] R. Riggio, A. Bradai, D. Harutyunyan, T. Rasheed and T. Ahmed, "Scheduling wireless virtual networks functions," *IEEE Transactions on network and service management,* vol. 13, no. 2, p. 240–252, 2016.

[126] C. Papagianni and J. Baras, "Rethinking service chain embedding for cellular network slicing," in *IEEE IFIP Networking*, 2018.

[127] A. Patel, M. Vutukuru and D. Krishnaswamy, "Mobility-aware VNF placement in the LTE EPC," in *IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, 2017.

[128] D. Dietrich, C. Papagianni, P. Papadimitriou and B. J.S., "Network function placement on virtualized cellular cores," in *9thInternational Conference on Communication Systems and Networks (COMSNETS)*, 2017.

[129] A. Baumgartner, V. Reddy and T. Bauschert, "Combined virtual mobile core network function placement and topology optimization with latency bounds," in *4th European Workshop on Software Defined Networks*, 2015.

[130] S. Sahhaf, W. Tavernier, M. Rost, S. Schmid, D. Colle, M. Pickavet and P. Demeester, "Network service chaining with optimized network function embedding supporting service decompositions," *Computer Networks,* vol. 93, p. 492–505, 2015.

[131] R. Cohen, L. Lewin-Eytan, J. Naor and D. D. Raz, "Near optimal placement of virtual network functions," in *IEEE Conference on Computer Communications (INFOCOM)*, 2015.

[132] D. Dietrich, A. Abujoda, A. Rizk and P. Papadimitriou, "Multi-providerservice chain embedding with nestor," *IEEE Transactions on Networkand Service Management,* vol. 14, no. 1, p. 91–105, 2017.

[133] R. Shi, J. Zhang, W. Chu, Q. Bao, X. Jin, C. Gong, Q. Zhu, C. C. Yu and S. S. Rosenberg, "MDP and machine learning-based cost-optimization of dynamic resource allocation for network function virtualization," *IEEE International Conference on Services Computing,* pp. 65-73, 2015.

[134] A. Pietrabissa, S. Battilotti, F. Facchinei, A. Giuseppi, O. Guido, M. Panfili and V. Suraci, "Resource management in multi-cloud scenarios via reinforcement learning," in *34th Chinese Control Conference (CCC)*, 2015.

[135] F. Liberati, "Service mapping," TNOVA project, 2016.

[136] L. Wang and E. Gelenbe, "Adaptive dispatching of tasks in the cloud," *IEEE Transactions on Cloud Computing,* vol. 6, p. 33–45, January 2018.

[137] J. Koo, V. B. Mendiratta, M. R. Rahman and A. Walid, "Deep Reinforcement Learning for Network Slicing with Heterogeneous Resource Requirements and Time Varying Traffic Dynamics," in *15th International Conference on Network and Service Management (CNSM)*, 2019.

[138] Y. Xiao, Q. Zhang, F. Liu, J. Wang, M. Zhao, Z. Zhang and J. J. Zhang, "NFVdeep: Adaptive online service function chain deployment with deep reinforcement learning," in *International Symposium on Quality of Service (IWQoS)*, 2019.

[139] C.-Y. Chang, N. Nikaein, O. Arouk, K. Katsalis, A. Ksentini, T. Turletti and K. Samdanis, "Slice Orchestration for Multi-Service Disaggregated Ultra-Dense RANs," *IEEE Communications Magazine,* vol. 56, no. 8, pp. 70-77, Auguat 2018.

[140] D. De Vleeschauwer, J. Baranda, J. Mangues-Bafalluy, C. F. Chiasserini, M. Malinverno, C. Puligheddu, L. Magoula, J. Martín-Pérez, S. Barmpounakis, K. Kondepu, L. Valcarenzhi, X. Li, C. Papagianni and A. Garcia-Saavedra, "5Growth Data-Driven AI-Based Scaling," in *Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit)*, 2021.

[141] T. Lorido-Botran, J. Miguel-Alonso and J. A. Lozano, "A review of auto-scaling techniques for elastic applications in cloud environments," *Journal of grid computing,* vol. 12, no. 4, pp. 559-592, 2014.

[142] M. Z. Hasan, E. Magana, A. Clemm, L. Tucker and S. L. D. Gudreddi, "Integrated and autonomic cloud resource scaling," in *IEEE network operations and management symposium*, 2012.

[143] E. Casalicchio and L. Silvestri, "Autonomic management of cloud-based systems: the service provider perspective," in *Computer and Information Sciences III*, London, Springer, 2013, pp. 39-47.

[144] R. Han, L. Guo, M. M. Ghanem and Y. Guo, "Lightweight resource scaling for cloud applications," in *12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012)*, 2012.

[145] T. Lorido-Botran, J. Miguel-Alonso and J. A. Lozano, "A review of auto-scaling techniques for elastic applications in cloud environments," *Journal of grid computing,* vol. 12, no. 4, pp. 559-592, 2014.

[146] Q. Zhu and G. Agrawal, "Resource provisioning with budget constraints for adaptive applications in cloud environments," *IEEE Transactions on Services Computing,* vol. 5, no. 4, pp. 497-511, 2012.

[147] H. C. Lim, S. Babu, J. S. Chase and S. S. Parekh, "Automated control in cloud computing: challenges and opportunities," in *1st workshop on Automated control for datacenters and clouds*, 2009.

[148] S. M. Park and M. Humphrey, "Self-tuning Virtual Machines for Predictable eScience," in *9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, 2009.

[149] T. L. Duc, R. G. Leiva, P. Casari and P. O. Östberg, "Machine learning methods for reliable resource provisioning in Edge-cloud computing: A survey," *ACM Computing Surveys (CSUR) ,* vol. 52, no. 5, 2019.

[150] J. Martín-Pérez, K. Kondepu, D. De Vleeschauwer, R. V., Guimarães, S. A. C. and C. J. Bernardos, "Dimensioning of V2X Services in 5G Networks through Forecast-based Scaling," preprint arXiv:2105.12527, 2021.

[151] S. Lange, H. G. Kim, S. Y. Jeong, H. Choi, J. H. Yoo and J. W. K. Hong, "Machine learning-based prediction of vnf deployment decisions in dynamic networks. In 2019," in *20th Asia-Pacific Network Operations and Management Symposium (APNOMS '19)*.

[152] S. Rahman, T. Ahmed, M. Huynh, M. Tornatore and B. & Mukherjee, "Auto-scaling VNFs using machine learning to improve QoS and reduce cost," in *2018 IEEE International Conference on Communications (ICC)* , 2018.

[153] D. Lee, J. H. Yoo and J. W. K. Hong, "Deep Q-networks based auto-scaling for service function chaining," in *16th International Conference on Network and Service Management (CNSM'20)*, 2020.

[154] Z. Zhou, T. Zhang and A. Kwatra, "NFV closed-loop automation experiments using deep reinforcement learning," in *IEEE Conference on Computer Communications INFOCOM 2019*, 2019.

[155] P. Gabriela, D. Lee, N. Van Tu and J. W. K. Hong, "Machine Learning-Based Auto-Scaler for Video Conferencing Systems," in *7th International Conference on Network Softwarization (NetSoft'21)*, 2021.

[156] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare and D. Hassabis, " (2015). Human-level control through deep reinforcement learning," *Nature,* vol. 518, no. 7540, pp. 529-533, 2015.

[157] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch and G. Caire, "FemtoCaching: Wireless content delivery through distributed caching helpers," *IEEE Trans. Inf. Theory,,* vol. 59, no. 12, p. 8402–8413, 2013.

[158] A. Asheralieva and D. Niyato, "Combining contact theory and Lyapunov optimization for content sharing with edge caching and device-todevice communications," *IEEE/ACM Trans. Netw.,,* pp. vol. 28, no. 3, pp. 1213–1226, 2020.

[159] G. Paschos, A. Destounis and G. Iosifidis, "Online convex optimization for caching networks," *IEEE/ACM Trans. Netw.,* pp. vol. 28, no. 2, pp. 625–638, 2020.

[160] E. Leonardi and G. Neglia, "Implicit coordination of caches in small cell networks under unknown popularity profiles," *IEEE JSAC,* p. 2018..

[161] L. Chen, L. Song, J. Chakareski and J. Xu, "Collaborative content placement among wireless edge caching stations with time-to-live cache," *IEEE Trans. Multimedia,* pp. vol. 22, no. 2, pp. 432–444, 2020.

[162] D. Carra, G. Neglia and P. Michiardi, "Elastic provisioning of cloud caches: A cost-aware TTL approach," *IEEE/ACM Trans. Netw.,* pp. vol. 28, no. 3, pp. 1283–1296, 2020.

[163] M. Dehghan and et al., "A utility optimization approach to network cache design," *IEEE/ACM Trans. Netw.,* pp. vol. 27, no. 3, pp. 1013–1027, 2019.

[164] B. N. Bharath, K. G. Nagananda and H. V. Poor, "A learning-based approach to caching in heterogenous small cell networks," *IEEE Transactions on Communications,* vol. 64, no. 4, p. 1674–1686, 2016.

[165] J. Xiong, Y. Fang, P. Cheng, Z. Shi and W. Zhang, "Distributed caching in converged networks: A deep reinforcement learning approach," *IEEE Transactions on Broadcasting,* vol. 67, no. 1, pp. 201-211, March 2021.

[166] A. Sadeghi, F. Sheikholeslami, A. Marques and G. B. Giannakis, "Reinforcement learning for adaptive caching with dynamic storage pricing," *IEEE Journal on Selected Areas in Communications,* vol. 37, no. 10, p. 2267–2281, 2019.

[167] A. Sadeghi, F. Sheikholeslami and G. B. Giannakis, "Optimal and scalable caching for 5G using reinforcement learning of space-time popularities," *IEEE Journal of Selected Topics in Signal Processing,* vol. 12, no. 1, p. 180–190, 2018.

[168] S. O. Somuyiwa, A. György and D. Gündüz, "A Reinforcement-learning Approach to Proactive Caching in Wireless Networks," *IEEE Journal on Selected Areas in Communications,* vol. 36, no. 6, p. 1331–1344, 2018.

[169] X. Wang, Z. Zhou, F. Xiao, K. Xing, Z. Yang, Y. Liu and C. C. Peng, "Spatio-temporal analysis and prediction of cellular traffic in metropolis," *IEEE Transactions on Mobile Computing,* vol. 18, no. 9, p. 2190– 2202, 2019.