Università
di Genova

PhD Program in Science and Technology for Electronic and Telecommunication Engineering

Power Measurements and Saving Techniques in Beyond 5G Networks

Beatrice Siccardi

Thesis submitted for the degree of *Doctor of Philosophy* (38° cycle)

March 27th 2026

Prof. Raffaele Bolla	Tutor
Prof. Roberto Bruschi	Co-tutor
Prof. Maurizio Valle	Head of the PhD program

Università di Genova
DITEN DIPARTIMENTO
DI INGEGNERIA NAVALE, ELETTRICA,
ELETTRONICA E DELLE TELECOMUNICAZIONI

Power Measurements and Saving Techniques in Beyond 5G Networks

Beatrice Siccardi

⋈

Committee:

Reviewers

Fabrizio Granelli, Full Professor, University of Trento, Italy
Rosario Garroppo, Associate Professor, University of Pisa, Italy

Tutors

Raffaele Bolla, Full Professor, University of Genoa, Italy
Roberto Bruschi, Full Professor, University of Genoa, Italy

University of Genoa

Beatrice Siccardi

Power Measurements and Saving Techniques in Beyond 5G Networks

xvi+108 p.

Abstract

The rise of 5G systems and Beyond 5G (B5G), along with state-of-the-art cloud and software engineering solutions, have brought to a radical shift of networking technologies with respect to the previous generations. However, the increased performance and flexibility have come at the cost of reduced sustainability, mainly from an environmental standpoint. Infrastructure owners can introduce power management solutions in their datacenters, but the improvement is negligible unless tenants operating on top of the physical infrastructure are enabled and incentivized to customize their slices and their applications towards more power efficient and carbon neutral profiles.

For these reasons, the thesis addresses the power efficiency issue. In further detail, herein, I present a flexible Observability, Monitoring and Analytics framework whose main targets revolve around measuring the power consumption of B5G Network Functions (NFs) and the application of algorithms (also involving Artificial Intelligence (AI)) to promote power efficiency.

The crucial part of the aforementioned framework is the Management Data Analytics Function (MDAF) whose main task is to provide a power measurement of B5G NFs which takes into account the role virtualization plays (i.e., the sharing of physical resources). Tests are performed in a 5G network composed of opensource softwares and commercial products. Through a statistical analysis, Scaphandre, a well-known power monitoring tool in virtualized environments, is found to be the most appropriate tool of comparison for the MDAF. Results show that the MDAF is able to detect a higher power consumption than Scaphandre which could become significant in case of NFs deployed in multiple instances. Moreover, the difference in power is mainly ascribable to operations related to the container management orchestrator and the scheduler of the Operating System (OS).

One of the two power saving techniques presented in this thesis concerns the exploitation of the well known power saving technique commonly used in general-purpose server: Low Power Idle (LPI), which consists in putting to (various degrees of) "sleep" Central Processing Units (CPUs) when not in use. Tests are performed on a 5G User Plane Function (UPF) and its power consumption and performance (latency) are evaluated. Results show that, in most cases the deeper the "sleep" states are enabled, the more power is saved. Furthermore, results show that this power saving does not come at a performance cost since no packet losses are detected in any test.

Finally, concerning the Analytics part of the aforementioned framework, an Artificial Neural Network (ANN) -based algorithm is presented. Its main goal is to, via a multiclass classification problem, drive the NF scaling decisions. This algorithm uses a blend of both "black-box" Key Performance Indicators (KPIs) (e.g., CPU and memory utilization) and "white-box" KPIs (e.g., NF metrics, such as the number of Packet Data Unit (PDU) sessions, etc.). The model proposed herein is application- and implementation-independent since the white-box KPIs are extracted from 5G standardized interfaces. The proposed model is tested in a 5G environment and compared with a standard threshold-based one. Results show a 97~98% accuracy in the training and validation phases and an overall accu-

racy of more than 97% on never seen before samples and, thus, a decrease in computational resource usage.

Keywords: B5G, 6G, Observability, Sustainability, Energy Efficiency, Network Management, Machine Learning, MDAF, LPI, ACPI, ANN.

Contents

List of Figures

List of Tables

Acronyms

5G	Fifth-Generation
B5G	Beyond 5G
NF	Network Function
OpEx	Operational Expenditure
CapEx	Capital Expenditure
MNO	Mobile Network Operator
GHG	Green House Gas
BAU	Business-As-Usual
NFV	Network Function Virtualization
SDN	Software-defined Networking
AI	Artificial Intelligence
CPU	Central Processing Unit
RAM	Random Access Memory
MDAF	Management Data Analytics Function
ML	Machine Learning
MEC	Multi-access Edge Computing
ETSI	European Telecommunications Standards Institute
VM	Virtual Machine
OS	Operating System
eMbb	enhanced Mobile Broadband
mMTC	massive Machine Type Communications
URLLC	Ultra-Reliable Low-Latency Communication
SBA	Service-based Architecture
UPF	User Plane Function
DN	Data Network
AMF	Access and Mobility Management Function
UE	User Equipment

SMF	Session Management Function
PDU	Packet Data Unit
AUSF	Authentication Server Function
3GPP	Third Generation Partnership Project
NSSF	Network Slice Selection Function
NRF	Network Repository Function
API	Application Programming Interface
PCF	Policy Control Function
NEF	Network Exposure Function
UDM	Unified Data Management
AF	Application Function
UDF	Unstructured Data Storage Function
5G-EIR	5G Equipment Identity Register
PEI	Permanent Equipment Identifier
RAN	Radio Access Network
NS	Network Service
VNF	Virtual Network Function
CNF	Containerized Network Function
DevOps	Development and Operations
RAPL	Running Average Power Limit
LPI	Low Power Idle
ACPI	Advanced Configuration and Power Interface
vCPU	virtual CPU
NCP	Network Connectivity Proxy
KPI	Key Performance Indicator
QoE	Quality of Experience
K8s	Kubernetes
SLA	Service Level Agreement

IT	Information Technology
KSM	Kube State Metrics
GUI	Graphical User Interface
PSU	Power Supply Unit
NWDAF	Network Data Analytics Function
IDAF	Infrastructure Data Analytics Function
SARIMA	Seasonal Auto-regressive Integrated Moving Average
CRI	Container Runtime Interface
GPU	Graphics Processing Unit
OAI	Open Air Interface
SUT	System under Test
XDP	eXpress Data Path
DPDK	Data Plane Development Kit
eBPF	extended Berkeley Packet Filter
DL	Deep Learning
ANN	Artificial Neural Network
ZSM	Zero touch network and Service Management
ENI	Experiential Networked Intelligence
ReLU	Rectified Linear Unit
MLP	Multilayer Perceptron
RNN	Recurrent Neural Network
CNN	Convolutional Neural Network
DNN	Deep Neural Network

CHAPTER 1

Introduction

The Fifth-Generation (5G) and Beyond (B5G) of mobile radio networks and edge computing technologies are expected to constitute the second wave of the Data Revolution [10] and to play a key role in the digital transformation towards a new hyper-connected society. In order to satisfy the stringent demand of 5G-ready applications, and to fully exploit the available network and computing resources, the distribution of application components and Network Functions (NFs) from the cloud to the edge of the network will become common practice.

Studies indicate that the intrinsic distributed and pervasive nature of B5G and edge computing are going to cause a noticeable usage and deployment increase of computing resources, increasing the associated infrastructure Operational Expenditure (OpEx) and Capital Expenditure (CapEx), and, consequently, their carbon footprint and energy requirements [2][11][12][13][14]. Much higher numbers than in today's cloud scenarios are foreseen, which cannot be compensated only by better efficiency levels coming from an inertial evolution of the ecosystem.

Figure 1.1 reports some indications on how 5G energy consumption is affecting the OpEx of Mobile Network Operators (MNOs), sensibly increasing the energy consumption (and, therefore, the environmental impact) of the infrastructure. It can be noted that nowadays MNOs are among major energy consumers in their relative countries. The rise of edge computing will further affect the infrastructure and its impact on energy consumption and Green House Gas (GHG) emissions, leading to an edge-cloud continuum composed by a large number of public/private micro/small/medium datacenters.

In detail, cloud computing datacenters are dimensioned against the aggregated workload of hosted applications publicly available on the Internet, while in B5G, edge facilities should be dimensioned against the workload produced by locally-connected mobile users and their (edge) applications. Therefore, the edge part of the B5G continuum cannot benefit from workload aggregation, and it will sensibly affect the sustainability of the ecosystem.

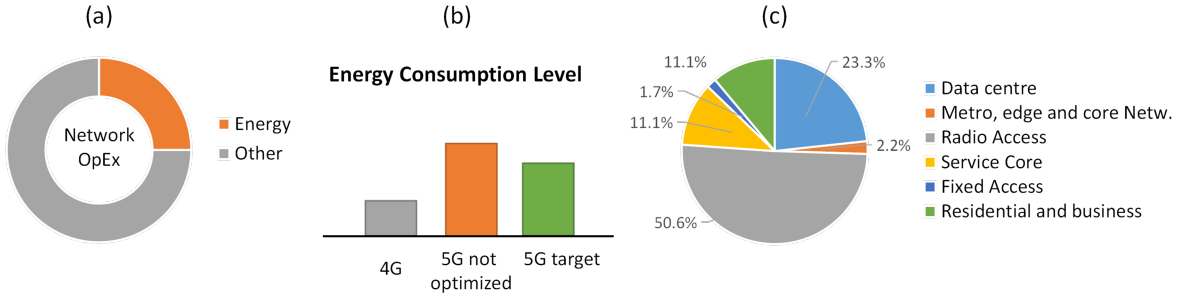

Figure 1.1: (a) contribution of energy consumption in the OpEx of MNOs (Source: GSMA 2019 [1], GSMA 2023 [2]]); (b) energy consumption levels of MNO facilities with only 4G, non-energy-optimized (Business-As-Usual – BAU) 5G, and energy optimized 5G (Source: GSMA 2019 [1]); (c) breakdown of energy consumption by network segment (source [3]).

To cope with this problem and to meet sustainable growth targets of both the ONU 2030 Agenda and the European Green Deal [15], the B5G continuum needs to rapidly evolve new foundation paradigms specifically addressing energy and carbon footprints of the overall ecosystem.

For all these reasons, the power consumption issue been a taken into consideration ever since 5G (e.g., 5G Advanced in 3GPP Rel. 18 [16]) and will be a crucial feature of B5G and 6G.

5G introduced, thanks to the concepts of Network Function Virtualization (NFV) and Software-defined Networking (SDN), the shift from a monolithic to a flexible structure. In further detail, the network components (i.e., the NFs) changed from physical devices to software applications. Clearly, this makes mobile networks more flexible and able to follow the As-a-Service paradigm. This is the starting point for applying algorithms (also including the use of Artificial Intelligence (AI)) that provide a B5G network which is always minimally dimensioned. Namely, power can be saved by providing only the B5G network services only when and where they are needed.

However virtualization also causes drawbacks. As a matter of fact, this evolution made the task of measuring the power consumption of a NF anything but straightforward. Nowadays, precisely measuring the power consumption of a NF, for instance through a power meter, is not possible. Moreover, these software applications (i.e., the NFs) are sharing the physical resources. Thus, measuring the power consumption of a NF entails the non trivial task of mapping physical resource (Central Processing Unit (CPU), Random Access Memory (RAM), disk, network I/O, etc.) power consumption to the hosted applications.

Moreover, the dematerialization of NFs and the possibility of having multiple slices on the same 5G network (each possibly devoted to a specific vertical application) further complicates the scenario by introducing multiple tenants. As a matter of fact, usually, the infrastructure owner will be different from the network owner and from the slice owner(s).

Thus, the aim of this thesis is to explore the following research questions:

- How can the power consumption of B5G networks be comprehensively measured?
- Can classic and/or newer methods be used to improve power efficiency in B5G networks?

The thesis includes four contributions: the former two (Chapters 3 and 4) tackle the first research question, while the latter two (Chapters 5 and 6) tackle the second one. The thesis starts by proposing a monitoring and observability framework whose main aim is to measure the power consumption of 5G NFs. In order to do so, bridging the gap between the multiple tenants is fundamental. As a matter of fact, measuring the power consumption necessarily starts with the infrastructure layer and the power consumption of the physical components; then, this needs to be linked with the upper layers (network and slice) in order to have a comprehensive power consumption. Afterwards, any kind of analytics can be applied ranging from statistical analysis to methods involving AI. In detail, in this thesis, two topics are explored to improve the power efficiency. The former explores a classic general-purpose server power saving techniques and applies it to a 5G network. While, the latter exploits a Machine Learning (ML) technique to drive the decision on NF scaling in order to provide the 5G services only when actually needed.

The remainder of the thesis is organized as follows. Chapter 2 introduces the topics that are going to be tackled in later chapters. Chapter 3 introduces a new Observability, Monitoring and Analytics Framework whose design and implementation is general and power-efficiency focused; herein common monitoring power tools are analysed and compared through a statistical analysis as well. Then, in Chapter 4, the aforementioned framework is applied to a 5G network, namely each component is mapped to a specific NF and a prototype of Management Data Analytics Function (MDAF) is presented. The proposed MDAF plays a fundamental role in appropriately measuring the power consumed by 5G NFs. After tackling the power consumption measurement issue, the thesis focuses on how power consumption can be reduced. Thus, Chapters 5 and 6 present and analyse two methods to improve the power efficiency of a B5G network. The former exploits a power saving techniques commonly employed in general-purpose servers, while, the latter a ML technique. Finally, conclusions are drawn in Chapter 7.

CHAPTER 2

Previous work

In this chapter, state-of-the-art methods necessary to address the problems tackled in this thesis are briefly reviewed.

First, a quick overview on 5G and its main features is presented. Then, an overview of the topics used in the thesis is introduced. These include energy efficiency in B5G networks, the most common power saving techniques for general-purpose servers and NF scaling.

2.1 5G and Beyond

Modern cloud technologies and architectures are largely recognized as the foundations of the upcoming 5G ecosystem [17] [18]. These technologies are expected to not only provide the needed means for enabling the softwarisation revolution in telecommunication infrastructures (mainly through the NFV framework), but also to act as key enablers for new (more pervasive and more network-integrated) computing paradigms, like, for instance, Fog and Multi-access Edge Computing (MEC) [19].

Virtualization allows to share high computational, storage and networking resources among different tenants; in particular this provides both the aggregation of resources in central data centers and the isolation among the different tenants. The 5G System architecture is defined to support data connectivity and services enabling deployments to exploit state-of-the-art NFV and SDN technologies. NFV is a framework developed by the European Telecommunications Standards Institute (ETSI) NFV Industry Specification Group. Its definition is [20]: "NFV aims to transform the way that network operators architect and operate networks and network services by evolving standard IT virtualization technology to consolidate many network equipment types onto industry standard high volume servers, switches and storage."

NFV virtualizes the NFs carried out by proprietary hardware appliances such as switches, routers, firewalls, etc; these devices become software applications. Through

NFV, a more flexible and more reliable network can be provided since virtual functions can be moved from a physical server to another. Furthermore, NFV reduces costs needed to put devices on the market. Being 5G NFs software application, they can be developed in two forms: Virtual Machines (VMs) or containers. The former correspond to Hardware being virtualized; VMs are software programs that emulate the functionality of a physical hardware or computing system containing their own operating system and applications/services. The latter, on the contrary, correspond to the Operating System (OS) being virtualized: containers create isolated OS environments within the same host system kernel; this means that different containers share the kernel, but each has its own files and processes. Containers are considered as a lightweight virtualization as compared to virtual machines [21]. Moreover, containerization allows the shift from monolithic applications to microservice-based applications [22] where an application is split into several loosely coupled microservices. These applications are, therefore, more easily scalable and more resistant to failure for two reasons: the former is that, since microservices are small, they are faster to start or restart in case of failure, the latter is that, being loosely coupled among each other, the failure of a microservice will not affect the others.

5G needs to be flexible and tailored to the application since it supports services that have different requirements. 5G can provide a different service based on the application because of three reasons: it's service-based, it enables network slicing and it uses different bandwidths.

Figure 2.1 shows the three main service categories that 5G should support:

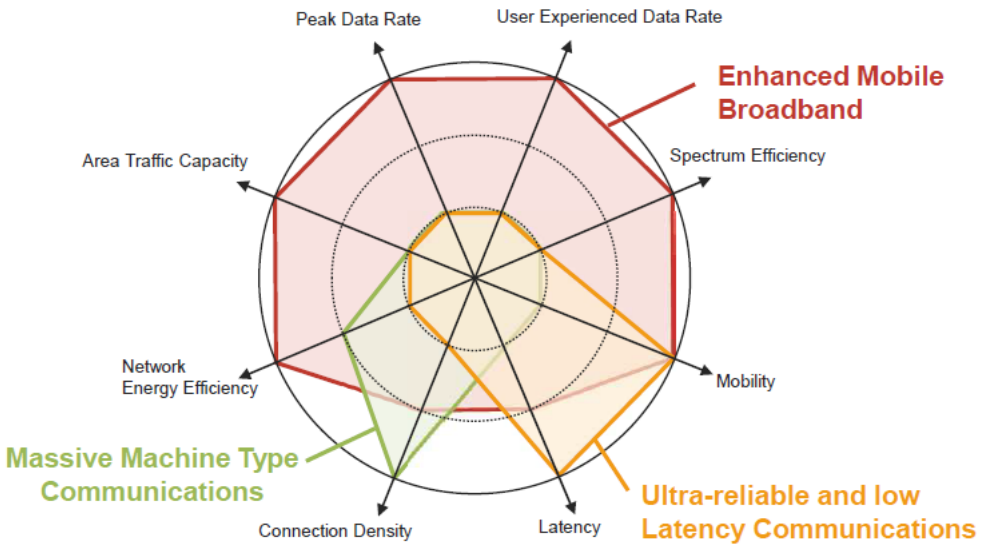

Figure 2.1: The 5G Network Service Categories [4].

- enhanced Mobile Broadband (eMBB) includes services similar to the ones provided by 4G, but with enhanced bandwidth, mobility, flexibility, etc.
- massive Machine Type Communications (mMTC) includes Iot applications that rely on a lot of sensors. Therefore, these applications need low power and a dense connection.
- Ultra-Reliable Low-Latency Communication (URLLC) includes applications that need low latency and no errors such as self-driving cars and remote surgical operation.

As mentioned earlier, 5G has a Service-based Architecture (SBA): the network elements are defined as NFs instead of network entities and the control plane is separated from the data plane. NFs communicate among themselves through RESTful APIs, therefore they use HTTP and its methods: GET, PUT, POST, DELETE.

Figure 2.2 shows the 5G NFs [23]:

- The User Plane Function (UPF) forwards the data traffic towards the Data Network (DN) that can either be the Internet or not. In the latter case the data traffic can be forwarded to an edge data center (closer to the user equipment) where computational resources are allocated. This leads to the Edge Computing technology.

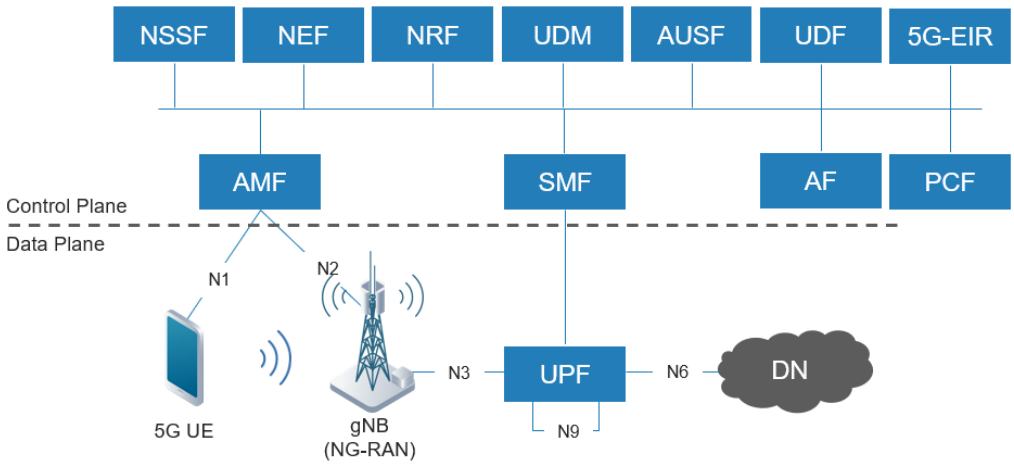

Figure 2.2: 5G Service-based Architecture.

- The Access and Mobility Management Function (AMF) deals with authentication, it transports messages between User Equipment (UE) and Session Management Function (SMF) and it can act as anchor point in case of handover.
- The SMF manages the lifecycle of the Packet Data Unit (PDU) sessions (instantiate, modify, delete).
- The Authentication Server Function (AUSF) supports the authentication for Third Generation Partnership Project (3GPP) access and untrusted non-3GPP access.
- The Network Slice Selection Function (NSSF) selects the set of Network Slice instances serving the UE.
- The Network Repository Function (NRF) provides registration and discovery functionality so that NFs can discover each other and communicate via Application Programming Interfaces (APIs).
- The Policy Control Function (PCF) provides policies (e.g., default type of PDU session, how to scale NFs, how to deal with the Network Exposure Function (NEF), etc.) to govern a slice of the network.
- The Unified Data Management (UDM) has two functionalities: it's a database and it provides some services (e.g., SMS).
- The Application Function (AF) is part of a vertical application that is considered trusted, therefore it can interact directly with 5G Core NFs.
- The NEF exposes the capabilities of the 5G slice to third-party applications functions. NEF can also receive instructions from third-party applications. Since NEF is used in case the third-party applications are not trusted (external to 5G Service providers), it must translate sensitive information before exposing them.
- The Unstructured Data Storage Function (UDF) is an optional function that supports the storage and retrieval of information as unstructured data by any NF. It's needed to have cloud-native stateless NFs.
- The 5G Equipment Identity Register (5G-EIR) is an optional Network Function that supports the status check of the Permanent Equipment Identifier (PEI – e.g., to check that it has not been blacklisted).

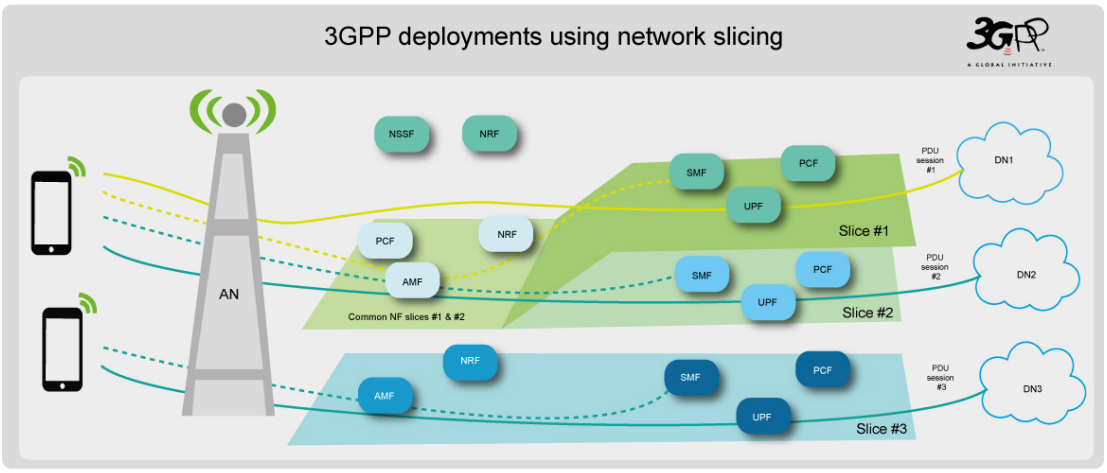

Figure 2.3: An example of Network Slicing [5].

Another key feature of 5G is network slicing that allows to create different instances of the same 5G network; all the instances share the same physical infrastructure and can be tailored to the vertical application. A UE can connect up to 8 network slices, and it has to have a single SMF for all the slices. A network slice can include both 5G Core Network and 5G Radio Access Network (RAN). Figure 2.3 shows an example of network slicing. In the picture one can notice that slice 1 and slice 2 share the AMF, PCF and NRF, while they don't share other NFs (SMF, UPF, PCF).

2.2 Energy efficiency in B5G networks

With the deployment of the 5G [24] radio mobile networks well underway, research interest is moving forward to the generation that will follow (B5G and 6G). While the widespread adoption of virtualization came with promises of improved energy efficiency thanks to the improved flexibility and scalability, in the reality the sustainability of 5G and B5G is causing serious concerns for industry and academia alike [11] [25] [13] [14].

Telecom providers are facilitating their embrace of cloud-native technologies. The transition from conventional communication applications to architectures based on microservices is set to simplify the creation of innovative Network Services (NSs), offering a significant degree of particularity. Nevertheless, the adoption of cloud-native technologies introduces new operational challenges. The efficient management of NSs needs innovative solutions for intricately monitoring and tracking widely distributed cloud-native NFs [26]. An important part of designing the B5G system is making wireless networks virtual,

allowing the easy adjustment of where and how much Virtual Network Functions (VNFs) are used based on specific needs. This flexibility is improved when VNFs are made as containers, called Containerized Network Functions (CNFs), and managed using a Development and Operations (DevOps) approach for continuous integration and deployment. Opting for containers over VMs allows for efficient and strong service applications built with a microservices structure [27].

As mobile generations continue to evolve and consumers need growth, the boundaries between the physical and digital worlds are fading. This trend has led to the deployment of ultra-dense networks, which poses significant challenges in terms of resource allocation and energy efficiency [28]. Consequently, the sustainability of mobile radio networks, 5G and B5G as cloud-native technologies, and edge cloud Computing is of paramount importance. Hence why the energy efficiency issue has been tackled ever since 5G (e.g., 5G Advanced in 3GPP Rel. 18 [16]) and will be the crucial point of B5G.

Energy efficiency in radio mobile networks has been interpreted as the ratio between the network capacity and its energy consumption. As such, increasing the energy efficiency ratio has been achieved with a significant raise of network capacities vs. a slight raise of energy requirements [1][29][30][31]. Furthermore, increasing the capacity of the networks, has always meant overprovisioning. Networks have always been dimensioned against their peak usage, resulting in a real utilization that is, most of the time, much lower than the capacity. The need for the capacity to dynamically follow the actual utilization is now apparent.

So far, the most energy-hungry part of the mobile networks has always been the RAN [32] which is composed of Base Stations (eNodeBs in 4G and gNodeBs in 5G) which allow the communication between the end user and the Core Network. However, this is expected to change with newer technologies. Real-time applications, and their strict requirements on latency, will be supported by 5G and B5G thanks to the deployment of such vertical applications either directly in the 5G/B5G network itself or in edge datacenters which will be geographically distributed worldwide in order to be closer to the end users and consequently reduce the overall latency. Therefore, the challenge will not only be located at the RAN but also at the edge.

So far, the energy consumption issue has been solved with the dynamic adaptation of server “sleep times” (i.e., low energy-consumption operating states) and/or server speed at wakeup, according to the traffic intensity and burstiness. 4G adopts the former solution. Therefore, the power consumption can be divided into a baseline which is always consumed and a part which depends on the number of active users. However, since devices still need

to perform some operations even when no users are active, as Malmodin measured in [32], the power consumed in "sleep mode" is significant: around 50% for 4G base stations.

Finally, another approach to the issue adopted thus far has been the utilization of renewable energy by Telecom Operators; this allowed a 8% reduction in operational emissions between 2019 and 2023, [33]. However, the advantage of renewable energy is less effective than in the case of datacenters because of the distributed nature of the telecommunication networks. This obstacle is going to be even greater in next generation technologies due to the edge-cloud continuum which is going to be composed of several smaller datacenters.

Thus, the key to reach a full environmental sustainability is to design mobile networks not bound to the network/computing capacity anymore, but rather to the real usage of resources. In this regard, 5G's architecture itself, being cloud-native and service-based, presents an opportunity as explained in Section 2.1. Due to their flexibility, 5/B5G networks can be provided only when, where and for the time needed by the end users. Moreover, they could also be moved based on the availability of renewable energy.

In order to dynamically shape the network to both fit the applications' needs and increase the energy efficiency, new metrics will be needed. Such information is not trivial to obtain because of virtualization and the design itself of future mobile technologies. Virtualization entails the difficult task of mapping hardware level metrics (i.e., energy consumption of CPU, RAM, etc.) to the hosted containers which share the same resources. Moreover, virtualizing the resources means that telecom operators do not necessarily own physical devices, but rather "rent" resources from infrastructure providers "as a Service" (i.e., only when actually needed). Therefore, two stakeholders (with different aims) are involved. Regarding the design, and, in particular, the link between B5G and edge computing, it is worth highlighting that a part or a whole vertical application could be hosted directly in the B5G network. Therefore, a third stakeholder should be considered. Finally, when considering the energy consumption of a mobile network, three stakeholders, acting at three different layers, are actually involved: the infrastructure provider, the telecom network operator and the vertical application. The energy related metrics should be measured and exposed at all the three layers. Once energy and application related information is available, the network can be tailored to fit the applications' needs while also keeping the environmental sustainability in mind. In this regard, AI is expected to play a key role [34].

Thus, to enhance the energy efficiency it is fundamental first to measure the power or energy itself of the B5G NFs. However, being B5G NFs deployed in the form of containers,

this task entails the mapping of shared physical resources (i.e., CPU, memory, disk, network I/O). Traditionally the CPU utilization, being CPU the most used hardware component, is considered to evaluate the energy/power consumption of virtualized components [35] or, alternatively, CPU hardware monitoring tools, like the ones made available by the Intel Running Average Power Limit (RAPL) [36] are exploited. One of the purpose of this thesis is to, on the one hand, compare the existing power/energy consumption monitoring tools and, on the other hand, present a more comprehensive method to measure the power consumption of B5G NFs.

2.3 General-purpose Server Power Saving Techniques

Reminding that B5G NFs run on general-purpose servers thanks to NFV, it is worth exploring the currently available energy/power saving techniques for servers.

The three strategies, depicted in Figure 2.4 [6], are the following:

1. Re-Engineering
2. Dynamic Adaptation
3. Smart Standby

Re-engineering approaches aim at introducing and designing more energy-efficient elements for network device architectures, at suitably dimensioning and optimizing the internal organization of devices, as well as at reducing their intrinsic complexity levels.

The dynamic adaptation of network resources is designed to modulate capacities of packet processing engines and of network interfaces, to meet actual traffic loads and requirements. This can be performed by using two power-aware capabilities, namely, power scaling and Low Power Idle (LPI) logic, which both allow the dynamic trade-off between packet service performance and power consumption.

Finally, smart standby approaches are used to smartly and selectively drive unused network portions to low standby modes, and to wake them up only if and when necessary. However, since today's networks and related services and applications are designed to be continuously and always available, standby modes have to be explicitly supported with special proxying techniques able to maintain the "network presence" of sleeping nodes/components. Applications of the smart standby approach to newer mobile technologies include the scaling (even to zero) of NF instances [37] [38]; this topic will be

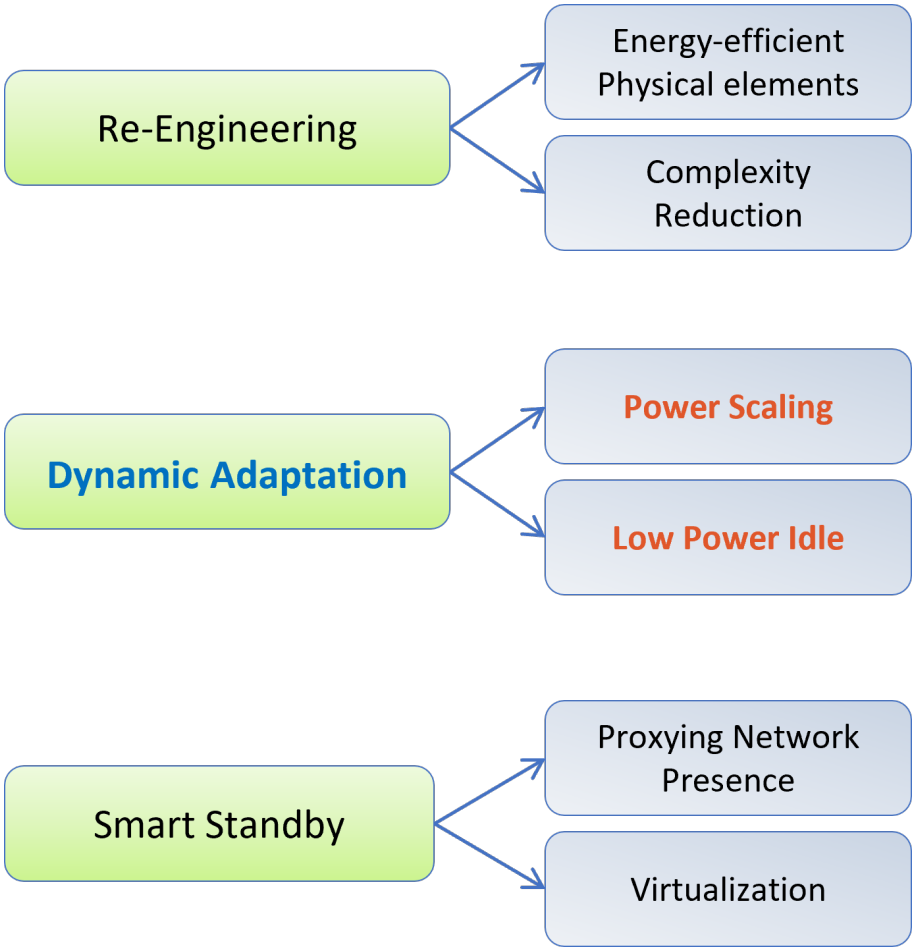

Figure 2.4: Energy efficiency strategies for computing systems [6].

explored in Section 2.4. It is worth noting that all these approaches are not mutually exclusive.

This section lays the foundation for Chapter 5 and thus focuses on the dynamic adaptation approach.

Dynamic adaptation approaches are aimed at modulating capacities of NF resources (e.g., link bandwidths, computational capacities of packet processing engines, etc.) according to current traffic loads and service requirements. Such approaches are generally founded on two main kinds of power management capabilities provided by the Hardware level, namely power scaling and idle logic.

In detail, power scaling capabilities allow dynamically reducing the working rate of processing engines or of link interfaces. This is usually accomplished by tuning the clock frequency and/or the voltage of processors, or by throttling the CPU clock (i.e., the clock

signal is gated or disabled for some number of cycles at regular intervals). Decreasing the operating frequency and the voltage of a processor, or throttling its clock, obviously allows the reduction of the power consumption and of heat dissipation at the price of slower performance. On the other hand, idle logic allows reducing power consumption by rapidly turning off sub-components when no activities are performed, and by re-waking them up when the system receives new activities. In detail, wake-up instants may be triggered by external events in a pre-emptive mode (e.g., new packet incoming or UE attaching), and/or by a system internal scheduling process (e.g., the system wakes itself up every certain time periods, and controls if there are new activities to process).

In general-purpose computing systems the Hardware implementation of both idle logic and performance scaling solutions is generally performed by pre-selecting a set of feasible and stable hardware configurations, which provide different trade-offs between power consumption and performance states. For example, different idle states are usually designed by selectively turning off an increasing number of hardware sub-elements. On one hand, this leads to reduce the power consumption during idle times; on the other hand, larger times are needed to wake up all the hardware sub-elements. In a similar way, power scaling hardware support is designed by pre-selecting a set of operating clock frequencies, whose values are sub-multiples of the maximum one and that provide silicon stability. Both these power-aware capabilities can be jointly adopted in order to adapt system performance to current workload requirements, and lead to different trade-offs between energy consumption and network performance.

The power scaling and LPI approaches were made available by the Advanced Configuration and Power Interface (ACPI) Specification standard published in 1996 [39]. The ACPI standard introduces two main different power saving mechanisms, namely performance and power states (P_x and C_x states), respectively, which can be individually employed and tuned for each core in the largest part of today's processors. The LPI approach leverages the C_x states; the C_0 power state is an active power state where the CPU executes tasks, while C_1 through C_N power states are processor "sleeping" or idle states, where the processor consumes less power. However, as the sleeping power state becomes deeper, the transition between the active (i.e., C_0) and the sleeping state (and vice versa) requires longer time. While in the C_0 state, ACPI allows the performance of the processor's core to be tuned through P_x state transitions. This empowers the power scaling approach. P_x states allow modifying the operating energy point of a core by altering the working frequency and/or voltage or throttling its clock. Therefore, they allow saving power while decreasing the performance of a processor.

The advantages and disadvantages of both approaches are depicted in Figures 2.5 and 2.6. In detail, Figure 2.5 shows the energy consumption depending on the P_x and C_x states while stressing on the role that the offered load plays: the higher the load the higher the performance provided by the core of the CPU needs to be (i.e., the higher the P_x). On the contrary, in case of lower loads, C_x states play a crucial role in saving energy.

In Figure 2.6 four scenarios are depicted: one where no power management techniques are applied, then, two where only LPI and only power scaling are applied, and finally one where both mechanisms are applied. On the one hand, LPI allows saving power when the CPUs are not used by going into "sleeping states" while keeping the highest performance (shown in red) when in active state. On the other hand, power scaling allows to decrease the power consumption also in active state at the drawback of both decreasing the performance and of consuming an active amount of power when the CPU is not processing anything.

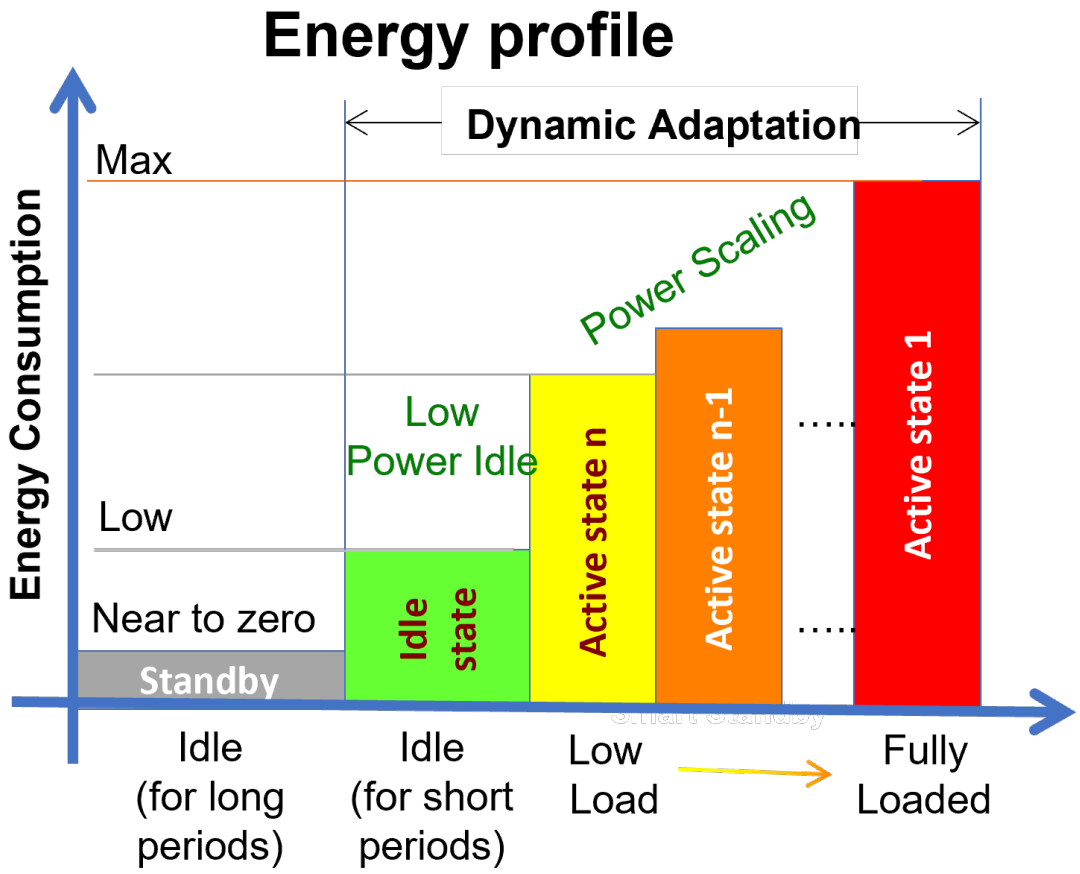

Figure 2.5: Energy consumption with respect to the offered load in standard operations and with LPI and power scaling [7].

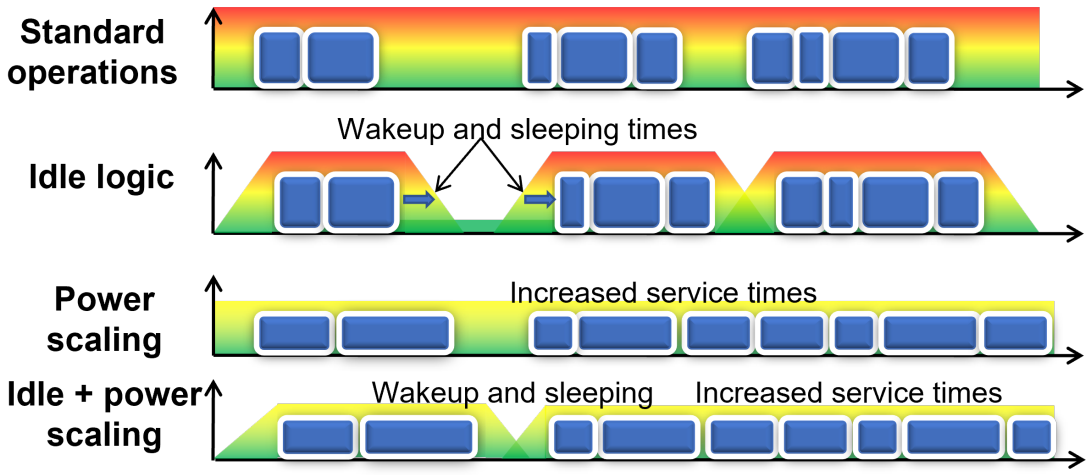

Figure 2.6: Packet service times and power consumptions in the following cases: standard operations, idle logic, power scaling, idle logic and power scaling [6].

Furthermore, Figure 2.6 highlights the impact the wakeup times (i.e., the time for the CPU core to transition from C_x to C_0) can have on peformance.

2.4 Network Function Scaling

Scaling is a well known mechanism to increase the capacity of a system (e.g., a physical computing system or a virtual one (VMs, containers)) commonly applied to cloud computing. This mechanism can assume two forms: vertical scaling and horizontal scaling. The former consists in increasing the resources of the computing system; for instance increasing the virtual CPUs (vCPUs) and memory available to a VM. While the latter consists in adding more instances of the same computing system; for instance creating a new instance of the VM. These mechanisms (horizontal scaling particurarly) are heavily used also in edge computing and 5G and Beyond.

Differently from cloud ones, edge computing facilities usually offer limited computing resources on highly heterogeneous hardware. In order to manage such resources as efficiently as possible, scaling mechanisms become crucial to guarantee the performance levels envisaged for 5G (and Beyond) applications. However, the proper design of such mechanisms is definitely a non-trivial task: along with the already mentioned limitations of edge datacenters, the lifecycle management of NSs and applications is performed on geographically distributed facilities, it must be fully automated and requires the injection

and update of applications/services (or parts of them) at run-time, only when, where and for the time needed by 5G/B5G end-users and connected things.

Horizontal scaling will play a crucial part in making the newer mobile technologies greener since the minimum number of instances of NFs can be made available only when and where necessary.

This rationale lays the foundation for "Edge Agility" which is a mechanism introduced by the 6Green project [40]. This feature should allow to move applications/slices at runtime across the distributed edge-cloud continuum based on events or forecasts. "Edge agility" is a natural extension of 5G, its foundation lays in the stateless essence of 5G NFs which enables the seamless relocation of their run-time session data and software images within the network and computing continuum (i.e., retrieving the external state of the microservice(s) to be moved, injecting it to new instances running in the new edge facilities, and properly remapping the network interconnectivity and data locations, etc.) [41]. Such relocations should be based on events and forecasts concerning B5G end-users (e.g., mobility, traffic intensity patterns, etc.) and the infrastructure (e.g., availability of renewable energy). Moreover, "Edge Agility" should significantly decrease the carbon footprint by "scaling to zero" the NFs and Vertical Applications when and where they are not needed. However, the aforementioned functions/applications should always be reachable, and, in case of need, they should be reactivated; in order to achieve this goal, sensing and network presence should be provided by B5G. Such a goal can be achieved by applying the well-known concept of Network Connectivity Proxy (NCP) [42] [43]: a new or extended microservice should handle network presence on behalf of scaled to zero functions/applications, redirect the traffic to the nearest active instances and reactivate them when needed.

Thus, scaling mechanisms become crucial to manage resources as efficiently as possible while still being able to guarantee the performance levels envisaged for B5G applications. Most works in the state of the art on scaling decisions use threshold-based solutions. In most of the threshold-based solutions, few Key Performance Indicators (KPIs) are analyzed to trigger the scaling decision. In particular, both [44] and [45] exploit only one metric, CPU utilization; in [46] CPU usage, RAM usage and Quality of Experience (QoE) are adopted. The ones which rely on AI and ML usually focus on traffic forecasting rather than on scaling decisions.

Finally, the possibility to deploy 5G NFs in the form of containers makes implemeting scaling mechanism even easier thanks to the feature of the Container Management Orchestrators (e.g., Kubernetes (K8s) [47]). An example of this can be found in [37].

CHAPTER 3

A Monitoring, Observability and Analytics Framework to Improve the Sustainability of B5G

3.1 Introduction

As mobile generations continue to evolve and consumers need growth, the boundaries between the physical and digital worlds are fading. This trend has led to the deployment of ultra-dense networks, which poses significant challenges in terms of resource allocation and energy efficiency [28]. Consequently, the sustainability of mobile radio networks, B5G as cloud-native technologies, and edge cloud computing is of paramount importance.

In this chapter, I focus on the environmental sustainability of the use-phase of general-purpose servers on which the B5G NFs are deployed. Thus, sustainability refers to achieving optimal resource utilization and energy efficiency while ensuring the seamless functioning of B5G technologies. Since, the presence of multiple network slices, available for rent by various parties, makes traditional monitoring and observability systems for energy consumption almost ineffective and the dynamic nature of MEC often leads to the simultaneous presence of both idle and overloaded servers [48]; the design of an appropriate monitoring and observability framework for B5G networks becomes essential.

Service monitoring has always been crucial for telecom operators to troubleshoot networks and ensure compliance with customer Service Level Agreements (SLAs). As telecom transitions to virtualized environments, real-time observability becomes even more critical as a key requirement for DevOps operations. In the context of B5G applications being implemented through microservices, the associated VNF components,

The content of this chapter has been published in the following paper:

M. Akbari, R. Bolla, R. Bruschi, F. Davoli, C. Lombardo and B. Siccardi, "A Monitoring, Observability and Analytics Framework to Improve the Sustainability of B5G Technologies," 2024 IEEE International Conference on Communications Workshops (ICC Workshops), Denver, CO, USA, 2024, pp. 969-975, doi: 10.1109/ICCWorkshops59551.2024.10615948.

packaged as containers, can be frequently initiated, stopped, updated, scaled, and migrated. A monitoring and observability system must possess the capability to adjust to alterations automatically and consistently in the deployment and communication of microservices. Consequently, current monitoring frameworks need to undergo evolution to meet the following criteria: adaptability, responsiveness, scalability, customization, manageability, distribution, isolation of multiple tenants, punctuality.

Contemporary Information Technology (IT) systems and following mobile networks have grown in complexity in four specific ways: modular (composition of different services and layers), distributed (computational and geographical), dynamic (MEC) and ephemeral (development pace in mobile networks). The rising complexity in this composite structure presents a significant challenge for traditional monitoring technology. These four dimensions of complexity indicate that IT systems are transforming to include a growing variety of diverse and independent components.

In the past, monitoring technologies were tailored to the characteristics of the components (e.g., networks, servers, storage and applications) being implemented separately and monitored [49]. The decision to specialize was influenced by two key factors. First, different components are generating distinct data types in specific scenarios, making it logical to customize data ingestion technologies based on the context of data collection. Second, these components are interacting infrequently and unpredictably, making it unnecessary to integrate observations from one component with those from another. This meant that troubleshooting for root cause analysis rarely required looking beyond the immediate context. Yet, the increasing complexity of IT infrastructure has diminished the original reason for specialization. The multitude of diverse components has exceeded the ability to monitor them individually. Architectural layers have multiplied, and the role and behavior of components within each layer, such as containers or non-containers, have undergone significant changes. Importantly, the principles for interpreting the self-describing data generated by one component cannot be universally applied to another.

Monitoring in static and monolithic applications presents a relatively straightforward challenge. However, in microservices, the complexity of connections among diverse components has significantly increased due to multiplication and differentiation. The interdependence between components has surprisingly strengthened. Consequently, the cause of a performance issue in one component often originates from a state change in other components. A monitoring approach tailored to a specific component would offer only local information, which might not be adequate for an operations team to identify the cause of a problem, let alone predict its recurrence in the future [50].

As distributed IT systems and application patterns rapidly evolve, the operations management domain has witnessed the emergence of various new terminologies and approaches. A particularly crucial concept among these is observability which is defined as "the ability to measure the internal state of a system only by its external outputs"[51]. These external outputs are primarily known as telemetry data such as logs, traces and metrics. Observability offers both high-level overviews of the system's health and detailed insights into implicit failure modes. In DevOps, monitoring and observability are often discussed together; while both aim to ensure system reliability, they possess a subtle distinction and, in reality, are interlinked. Fundamentally, observability does not substitute monitoring, nor does it make monitoring obsolete; rather, they complement each other. Monitoring focuses on tracking a system's health through a set of predefined metrics and logs, aiming to detect specific types of failures. However, in distributed systems, changes are dynamic and often occur simultaneously and regularly. This can lead to issues that do not fit the predefined categories, causing them to be missed by the monitoring system. Observability, on the other hand, builds upon monitoring and helps in understanding where and why a failure occurred, along with what triggered the issue. It is crucial to combine the two in order to provide a complete and wide framework.

The goal of this chapter is to present a monitoring, observability, and analytics framework for greener B5G networks. An example of analytics, a statical dependence method, is presented; its aim is to on one hand, identify the most suitable monitoring tool(s), and on the other hand, create a reference point for a future anomaly detection mechanism.

3.2 Methodology

In the following subsections the proposed monitoring, observability and analytics framework for B5G platforms, shown in Figure 3.1, is presented. The former subsection focuses on the monitoring and observability part, while the latter focuses on the analytics applied on the collected data.

3.2.1 Monitoring and Observability

First, it is crucial to measure the power and/or energy consumption of the B5G components (i.e., containers or VMs) in order to then be able to optimize it. As mentioned in Section 2.2 this is a non-trivial task. My framework is based on the Kubernetes platform, since B5G components (i.e., NFs) are usually deployed as containers and Kubernetes is one of the

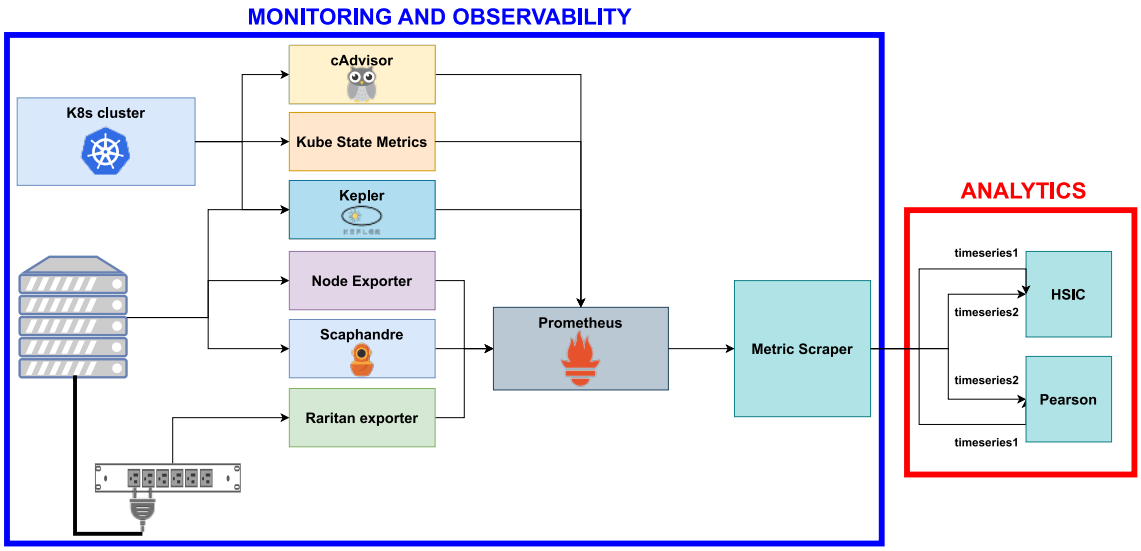

Figure 3.1: The proposed monitoring, observability and analytics framework.

most well-known platforms for managing containers. Nevertheless, this framework is quite modular and hence it can be modified in the monitoring, observability and analytics parts to achieve higher orders of scalability and flexibility. In the following, let me consider a simplified scenario: I do not actually deploy any B5G networks, but simply containerised applications. As shown in the blue encased part of Figure 3.1, the monitoring and observability part revolves around the data collection and minor data manipulation. The data is collected from two sources: the Kubernetes platform and the server machine itself. The data collection relies on well-known open-source applications which expose resource utilization and energy/power consumption of either processes (containers included) or of the physical server. Regarding Kubernetes, two exporters are used: cAdvisor [52] which provides the resource (i.e., CPU, memory, networking) usage of containers and Kube State Metrics (KSM) [53] which listens to the Kubernetes API server and generates metrics about the state of the objects (e.g., nodes, pods). Regarding the energy/power consumption, three applications are used: Kepler [54], Scaphandre [55] and the Raritan exporter [56]. Kepler provides the energy consumption of both the single containers/processes and of the whole servers. Scaphandre provides the same information, but in terms of power consumption. Both Kepler and Scaphandre rely on the information exposed by the Intel RAPL [36] counters. Moreover, the Raritan exporter provides the active power, temperature and other metrics offered by the Raritan Power Distribution Units to which the servers are plugged. Finally, Node Exporter [57] is used to provide resource (i.e., CPU, memory, networking)

usage at the server level. It is worth mentioning that ground-truth data for what concerns power/energy consumption of containers is difficult to obtain. Even exploiting the power consumption of the Power Distribution Units (provided by the Raritan exporter) to which the servers are connected is not sufficient, since that includes also environmental variables (e.g., temperature of the room) and all the additional processes executed by the OS itself in order to keep the server running and to support the virtualized environment. Thus, it is complicated to separate the containers' contribution from the actual power drawn from the PDUs. All the collected data is stored in a Prometheus [58] database. Prometheus is a time series database, that collects metrics enriched with a timestamp (i.e., representing the time instant in which Prometheus receives said metrics) and with labels which allow for better identification of the data. Prometheus collects information on a pull basis: it polls exporters every scraping interval (herein 10 seconds). Moreover, it provides metrics in four different data types: i) counter, that is a cumulative value which can only increase; ii) gauge, which is a value able to both decrease and increase; iii) histogram, which counts observations in configurable buckets; and iv) summary, similar to histogram samples observations, also provides a total count of observations and a sum of all observed values. Finally, the metrics saved on Prometheus are accessible either through its Graphical User Interface (GUI) or through its API. The Metric Scraper block shown in Figure 3.1 has the goal of scraping metrics from Prometheus and then add additional labels (e.g., pod name, namespace and so on obtained by KSM) to the Scaphandre ones.

3.2.2 Analytics

The analytics part of the proposed framework is shown in the red rectangle in Figure 3.1. This delves into the statistical dependence of timeseries, aiming to explore the correlation between resource utilization, power/energy consumption metrics, gathered from different monitoring tools and the actual metrics. However, it is worth highlighting that this is only one possible use of the collected data, and any analytics algorithm could be applied to the Prometheus-like metrics. Another possible analytics algorithm is presented in Chapter 6.

The concept of statistical dependence is used to describe a relation between any two features under study, such as individuals, objects, or environmental features. In the absence of deterministic dependence and independence, the statistical dependence remains as the characteristic trait of both features, showing the differences in the distributions of one variable for at least some of the levels of the other [59].

Statistical dependence and independence are fundamental concepts in statistics and ML and play a crucial role in various aspects of data analysis and ML tasks such as: feature selection [60], anomaly detection [61][62], dimensionality reduction [63], clustering [60] and optimization [64]. The two use-cases of statical dependence of interest in this article are feature reduction and anomaly detection. The former consists in identifying a subset of available features that is the most relevant for the data analysis of interest. In particular, herein, our goal is to choose the most appropriate application among Kepler and Scaphandre to measure the power/energy consumption. The latter, anomaly detection, consists in finding patterns in data that do not show an expected behavior and then trigger a warning/alert. When it comes to evaluating statistical dependency, numerous methods are prevalent in existing scientific literature. I focus on two: Pearson [65] and Hilbert-Schmidt Independence Criterion (HSIC) [66].

Pearson's index (ρ) stands out as the most widely employed measure of statistical dependence. It provides a comprehensive characterization of dependence in Gaussian scenarios and performs well in certain non-Gaussian situations. However, it is acknowledged to have limitations. Pearson's ρ is the ratio between the covariance of two variables and the product of their standard deviations:

$$\rho = \rho(X, Y) = \frac{\sigma(X, Y)}{\sigma_X * \sigma_Y} \tag{3.1}$$

where X and Y are two random variables with finite second moments. The result is a value between -1 and 1. Where -1 means negative dependence (i.e., if X increases then Y linearly decreases), 1 means positive dependence (i.e., if X increases then Y linearly increases as well) and 0 means independence. Finally, it is worth noting that Pearson is unreliable in case of non-linear dependence and in the presence of outliers.

HSIC was originally designed to evaluate the statistical dependence of the distribution-based Hilbert space embedding in statistical inference. Let us consider X as a reproducing kernel Hilbert space (RKHS) of functions from X to $\mathbb{R}$, where X is a separable metric space and $\mathbb{R}$ is the set of real numbers. To points $\mathbf{x}$, $\mathbf{x}'$ in X, there correspond elements $\varphi(\mathbf{x}), \varphi(\mathbf{x}') \in F$ (we refer to $\varphi : X \rightarrow F$ as feature mapping) such that

$$k(\mathbf{x}, \mathbf{x}') = \langle \varphi(\mathbf{x}), \varphi(\mathbf{x}') \rangle_F, \tag{3.2}$$

where $k : X \times X \rightarrow \mathbb{R}$ is an associated reproducing kernel.

Additionally, another RKHS G is also defined with respect to the separable metric space Z with feature mapping $\phi : Z \rightarrow G$ and the corresponding kernel is

$$l(\mathbf{z}, \mathbf{z}') = \langle \phi(\mathbf{z}), \phi(\mathbf{z}') \rangle_G, \tag{3.3}$$

where $\mathbf{z}, \mathbf{z}' \in Z$.The so-called HSIC is the square of the Frobenius or Hilbert–Schmidt norm of the cross-covariance operator:

$$\begin{aligned} HSIC\,(F,\ G,\ P_{xz}) = \ \| \ C_{xz} \ \|^2_{HS} = \ E_{x,x',\ z,z'}\ [k\ (x,\ x')\ l\ (z,\ z')] \\ -2E_{x,z}\ [E_{x'}\ [k\ (x,\ x')]\ E_{z'}\ [l\ (z,\ z')]] + E_{x,x'}[k(x,\ xı)]Ez, z'[l(z, z')] \end{aligned} \tag{3.4}$$

where $(\mathbf{x}, \mathbf{z})$ on $X \times Z$ are random variables jointly drawn from probability distribution P_{xz}, the cross-covariance operator between φ and ϕ can be defined as linear operator $C_{xz} : G \rightarrow F$, and $E_{x,x',\ z,z'}$ is the expectation on both $(\mathbf{x},\ \mathbf{z}) \sim P_{xz}$ and $(\mathbf{x}ı,\ \mathbf{z}ı) \sim P_{xz}$. It has been demonstrated that when F and G are RKHSs with characteristic or universal kernels [60] [66] [67] such as Laplace kernel and Gaussian kernel, $\mathbf{x}$ and $\mathbf{z}$ are statistical independent if and only if $HSIC(F, G, P_{xz}) = 0$ [66]. The larger the index, the stronger the dependence between two random variables. Moreover, this property allows the detection of nonlinear statistical dependence.

3.3 Experimental validation

3.3.1 Experimental Setup

The experiment's aim is to conduct measurements on a cloud-native scenario that mimics the nature of a B5G network albeit much simpler. Therefore, the setup consists of two physical servers which have all the applications shown in Figure 3.1 installed. All of them either run on Kubernetes or Docker except for Scaphandre which must run directly on the machine. In detail, one serves as the Kubernetes master node, and the other one as the worker node (i.e., where the actual workload runs).

Two tests were carried out exploiting iperf [68] which creates TCP or UDP streams and sends traffic from one host to another. In both tests, the server and the client were deployed as Kubernetes pods on the worker node. In the first test, the client is up and running for the whole duration, while in the second the client is deleted and redeployed every 1 hour. Each test lasted 24 hours.

3.3.2 Data Preprocessing

As mentioned in Section 3.2.1, the data are collected by a Prometheus database. However, it is worth mentioning that being pull-based, Prometheus has the drawback of having timeseries inconsistent in length. That occurs because if the exporter does not reply with the data whenever Prometheus polls it, no entry is saved in Prometheus. In the experiments, I found that cAdvisor was the exporter with the most issues. The inconsistency is shown in Figure 3.2 which displays a 3-minute plot reporting the number of samples for a Kepler metric and a cAdvisor one. It is worth noting that, while for Kepler we have a metric every 10 s, the same does not hold for cAdvisor.

This issue was resolved in two ways: up sampling or down sampling. The former adds missing samples through interpolation. The latter removes samples and keeps the value closest to the timestamp. While the latter will result in lower computation time, the former could have higher accuracy at the cost of inventing new samples.

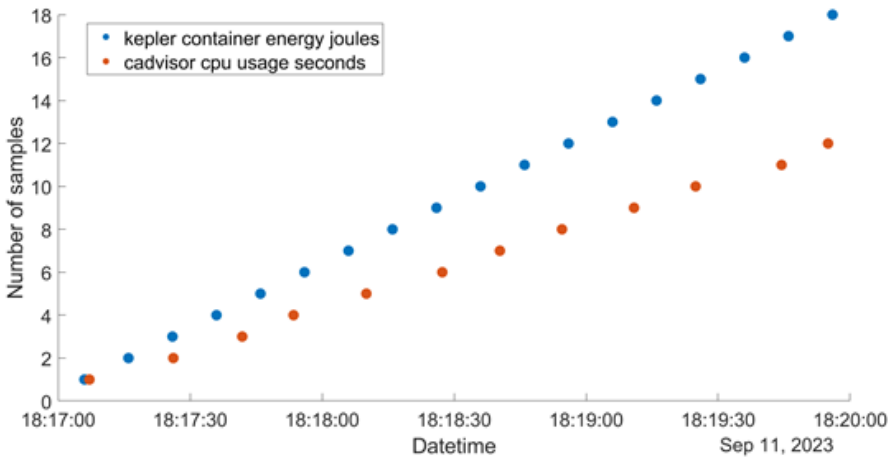

Figure 3.2: Inconsistency in length of the metrics.

3.3.3 Experimental Results

Herein, the Pearson and HSIC results are reported. However, before reporting the results it is worth mentioning the expected outcomes. In detail, I expect to find a positive dependency between the resource (i.e., CPU, memory, networking) utilization and the energy consumption. Since the results in all the cases (i.e., first test up sampled, first test down sampled, second test up sampled, and second test down sampled) are similar, only

Table 3.1: Results for the first test up sampled case. Both the indexes (Pearson and HSIC) and the resulting dependence/independence are reported. "/" indicates where the two dependence criteria give different results.

Compared metrics	Pearson's ρ	Pearson's p-value	HSIC's first index	HSIC's second index	Dependence result
Kepler container energy Container cpu usage seconds	0.9999	0	989.6	0.8026	Dependent
Kepler container energy Container network trasmit bytes	0.9999	0	989.9	0.8024	Dependent
Kepler container energy Container memory usage bytes bytes	0.9757	0	945.6	0.7917	Dependent
Kepler server energy Node cpu usage seconds bytes	0.002900	0.7874	0.1163	0.6550	Independent
Kepler server energy Node disk io time	0.003171	0.7681	0.1128	0.6550	Independent
Kepler server energy Raritan energy	0.003100	0.7731	0.1124	0.6556	Independent
Scaph container energy Container cpu usage seconds	0.9999	0	987.5	0.8025	Dependent
Scaph container energy Container network transmit bytes	0.9999	0	986.9	0.8023	Dependent
Scaph container energy Container memory usage bytes bytes	-0.09479	2.969E-17	6.842	0.6169	/
Scaph server energy Node cpu usage seconds	0.9998	0	1082	0.8033	Dependent
Scaph server energy Node disk io time	0.9999	0	1085	0.8041	Dependent
Scaph server power Raritan activepower	0.9033	0	987.6	0.8024	Dependent
Kepler container energy Scaph container energy	0.9999	0	987.6	0.8024	Dependent
Kepler server energy Scaph server energy	0.003924	0.7154	0.1101	0.6557	Independent

those for the first test up sampled cased for the iperf server are provided in Table 3.1. The metrics starting with container belong to cAdvisor, while the ones starting with node belong to Node Exporter. For each comparison four numbers are provided: for Pearson ρ and the p-value are provided, while for HSIC two indexes are provided. The two timeseries are dependent if the first index is larger than the second, otherwise they are independent. In all the comparisons, Pearson and HSIC report the same result even though in some cases the Pearson p-value is too high and therefore Pearson's result should be discarded. The only comparison in which the two criteria provide a different result is that between the Scaphandre energy consumption of the container and the cAdvisor memory usage of the container. Here, Pearson concludes for independence, while HSIC for low dependence (the difference among the two indexes is much lower than all the other cases).

Further examining the results, while Kepler successfully identifies the dependency between process resource utilization (cAdvisor) and the corresponding energy consump-

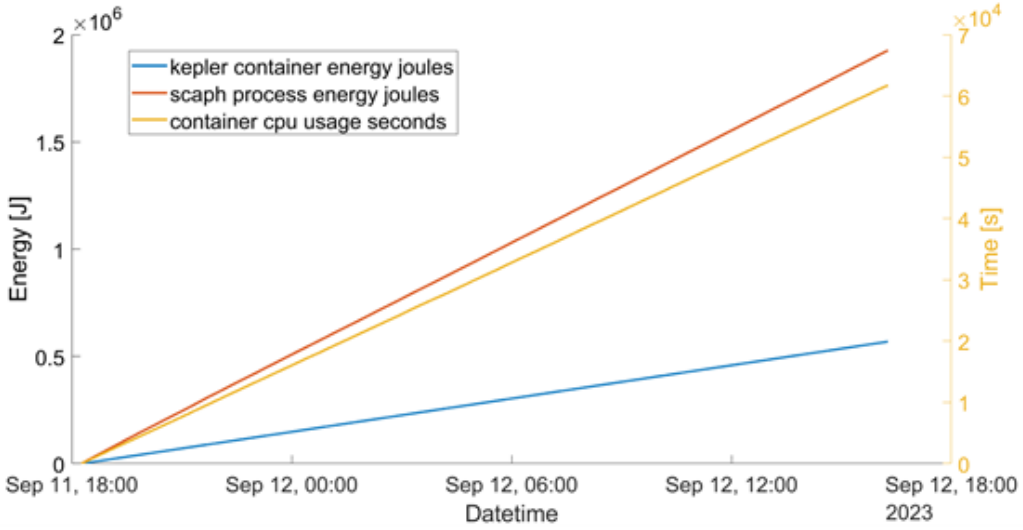

Figure 3.3: Scaphandre and Kepler energy measurements and cAdvisor CPU time over time.

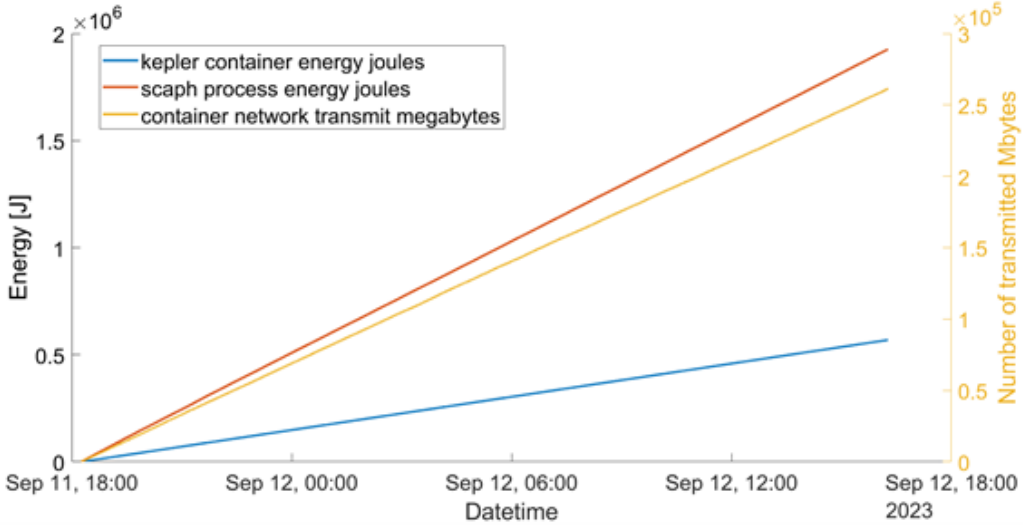

Figure 3.4: Scaphandre and Kepler energy measurements and cAdvisor CPU network transmission over time.

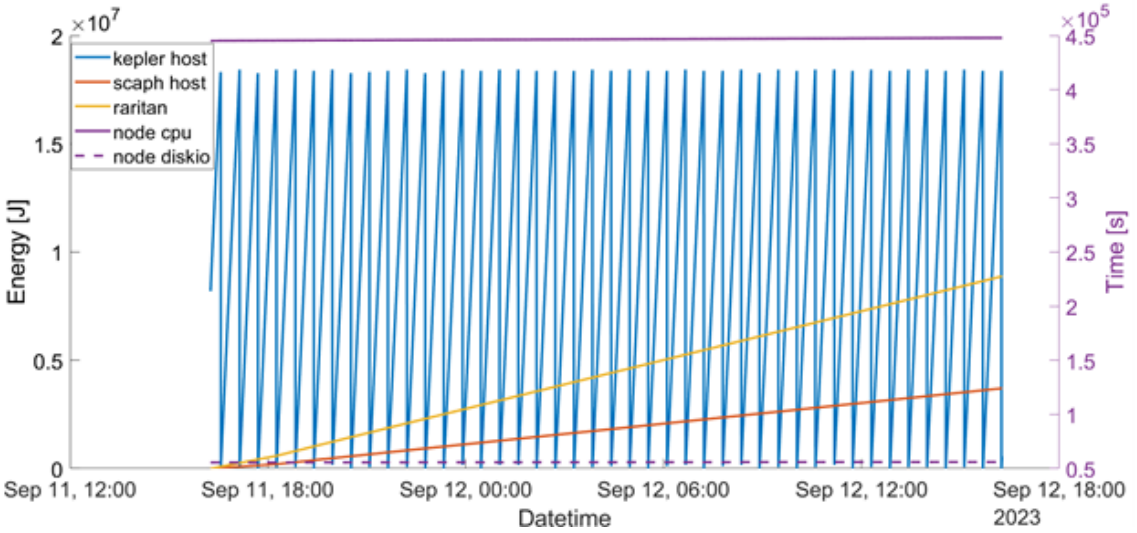

Figure 3.5: Node Exporter, Scaphandre, Kepler and Raritan server metrics over time.

tion, its efficacy diminishes when capturing the interdependence between server resource utilization (Node Exporter), PDU plug (Raritan), and their respective energy consumption. On the other hand, Scaphandre outperforms Kepler in discerning the relationship between host resource utilization (Node Exporter) and the energy consumption of both the server and the PDU plug (Raritan). Therefore, we conclude with Scaphandre being the better suited option for our framework.

For visualization purposes, the results are also reported in terms of figures. Figure 3.3 shows the container-level plots: the container energy consumption from Scaphandre and Kepler and the cAdvisor CPU usage time are reported. Figure 3.4 shows the relations between the energy consumption and the cAdvisor network transmission over time. In both figures, it is worth noting that all the timeseries display a positive dependence (i.e., when one increases, also the others increase).

Additionally, Figure 3.5 displays the server-level plots for Kepler, Scaphandre, Raritan and Node Exporter. As illustrated in Figure 3.5, there exists a positive dependency among Node Exporter server resource metrics, Kepler, Scaphandre host energy consumption, and Raritan Power Distribution Unit energy, all exhibiting an upward trajectory over time. Notably, a visible distinction emerges between Raritan PDU plug outgoing power and the measurements obtained from Scaphandre. This difference can be attributed to several factors: i) the omission of energy consumption data for other hardware components not measured by Scaphandre; ii) the active power supplied from the Power Supply Unit (PSU)

to the components; and iii) the energy dissipated by the electronics as heat. Another noteworthy consideration pertains to the behavior of the Kepler results: the plot shows an incremental pattern until reaching a threshold, after which they restart and resume increasing from zero. This is likely due to Intel RAPL counter resets on which Kepler is based.

Finally, it is worth mentioning the average computation times for both indexes: HSIC took 29.8 seconds (in the up sampled case) and 6.5 seconds (in the down sampled case), while Pearson took around 1.3 milliseconds (in the up sampled case) and 1.2 milliseconds (in the down sampled case). These results show that HSIC cannot be used at run-time (i.e., for anomaly detection). However, the computation times could be lowered if using a more performing server to run the two criteria. In detail, the Pearson index could be used at run-time to perform anomaly detection: when the results of Pearson correlation are not following the presented results, a warning can be triggered.

3.4 Conclusions

In this chapter I presented a monitoring, observability, and analytics framework for B5G platforms with a feature reduction and anomaly detection goal.

The proposed framework first tackled the issue of inconsistent length timeseries through preprocessing techniques, then it exploited statistical dependence criteria, specifically, the HSIC and Pearson methods. Pearson measures the normalized covariance between two variables, reflecting linear correlations, while HSIC provides a more complex structure suitable for non-linear cases. The complexity of HSIC that results in dramatic increase of computation time with respect to Pearson, makes it impractical for real time usage.

Results emphasize the importance of aligning the feature selection of the monitoring platform with resource utilization and choosing a suitable platform for power/energy consumption that accurately reflects actual energy usage. The investigation brings attention to variations in monitoring tools performance; Scaphandre tends to underestimate container memory power consumption while excelling in other metrics. In contrast, Kepler performs well in container resource metrics but falls short of expectations in host metrics. Although Scaphandre tends to underestimate container memory power consumption, its performance surpasses that of Kepler, making it more suitable for the monitoring and observability framework.

The framework proposed herein is going to be explored and improved further in the next Chapter.

Acknowledgements
This work has been partially supported by the Horizon Europe Research and Innovation Action 6Green (Grant Agreement no. 101096925), by the European Union under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, partnership on “Telecommunications of the Future” (PE00000001 - program “RESTART”) and “So-BigData.it – Strengthening the Italian RI for Social Mining and Big Data Analytics” – Prot. IR0000013.

CHAPTER 4

The Management Data Analytics Function (MDAF)

4.1 Introduction

As mentioned in Section 2.2, newer mobile network technologies are focusing more and more on power/energy efficiency. The first step towards such a goal, is to appropriately measure the power consumption of B5G network components. Among the main issues preventing the assessment of the power consumption ascribable to a specific user, to an application, or to a slice, is due to the fact that their virtualized components most of the times run on the same piece of hardware, making it hard to understand to what extent a specific VM or container is responsible for the power consumption of a server. Moreover, in a multi-domain ecosystem in which the infrastructure, the network platform and the vertical applications are owned by multiple separated entities [69], the task of inferring the consumption of a physical component ascribable to a virtual one is anything except trivial.

To this end, this chapter sets the observability, monitoring and analytics framework introduced in Chapter 3 in a 5G network: each block composing the framework is mapped to a specific 5G NF, and extends it. A crucial part of the framework is the MDAF, whose prototype is presented in the following. The MDAF was designed and developed within the

The content of this chapter has been published in the following papers:

M. Akbari, R. Bolla, R. Bruschi, C. Lombardo, N. S. Martinelli, B. Siccardi, "A Management Data Analytics Function for ethical 6G networks," Computer Networks, Volume 270, 2025, 111487, ISSN 1389-1286, https://doi.org/10.1016/j.comnet.2025.111487.

M. Akbari, R. Bolla, R. Bruschi, C. Lombardo, N. S. Martinelli and B. Siccardi, "Observe to Sustain – How to Enable Beyond 5G Networks to Target Sustainability Goals," 2024 IEEE 35th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), Valencia, Spain, 2024, pp. 1-7, doi: 10.1109/PIMRC59610.2024.10817341.

R. Bolla, R. Bruschi, C. Lombardo and B. Siccardi, "6G Enablers for Zero-Carbon Network Slices and Vertical Edge Services," in IEEE Networking Letters, doi: 10.1109/LNET.2023.3262861.

6Green Project [40], and it is part of a multi-domain observability framework introduced in [70].

The Observability Framework of B5G is composed of two main NFs: the Network Data Analytics Function (NWDAF) and the MDAF. While several articles in the literature can be found reading the former, this is not the case for the latter. In the following, I report, to the best of my knowledge, the MDAF-focused articles currently available in the literature. The authors in [71] present a scaling and load-balancing mechanism based on the MDAF; in detail the proposed MDAF monitors the CPU utilization of the AMF instances, performs analytics on such data and thus takes a scaling or a load-balancing decision. [72] presents an End-to-End Data Analytics Framework for 5G Architecture of which the MDAF is part; their implementation is particularly focused on the RAN. In [73] data usage control mechanisms to safeguard private network data when performing management data analytics are examined. In the PoC, the MDAF's goal is to analyse energy savings. Finally, authors in [74] delve into the role the MDAF has in building historical databases leveraged by AI mechanisms for autonomous control, management, and orchestration in 5G.

The MDAF proposed in the following is focused on power consumption: it represents an evolution to the 3GPP TS 28.533 definition [75], which already includes resource usage predictions, to extend the plethora of collected data to the energy context by interacting with the Infrastructure. Thanks to this interaction, the MDAF allows to quantify the portion of power consumed at bare-metal level ascribable to a VM or a container. As mentioned in Section 2.2, most of the works present in the literature consider only the CPU power consumption when it comes to the physical resources of servers. An example of such study is presented in [76] where the authors introduce an approach that measures the power consumption of different implementations of 5G networks (i.e., baremetal, VMs and containers) through the use of physical and software power meters. Concerning the latter, the authors only consider the well-known power measuring tool Scaphandre [55] which measures only the CPU power consumption. In this Chapter, I am proposing a method, embedded in the proposed MDAF, that takes into account all the physical components that are present in a server with a particular focus on CPU, RAM and networking components which are especially stressed in 5/B5G applications.

Results compare the proposed MDAF and Scaphandre, one of the most widespread open-source tools for collecting process-level power, when evaluating the consumption of several NFs. As explained in Chapter 3, Scaphandre was chosen because of its greater

ability to keep track of the resource (i.e., CPU, memory and networking) utilization than one of its alternative: Kepler.

4.2 Methodology

First, this section is devoted to tailoring the Observability, Monitoring and Analytics framework described in Chapter 3 to a B5G network. Then, each component of the framework is mapped onto the appropriate B5G and Infrastructure Function. Finally, the proposed MDAF is explained in details.

4.2.1 The Observability Framework

Although the focus of this chapter is on the MDAF, it is crucial to present its interactions with the other functions in the infrastructure and in the B5G Network domain. Therefore, the whole Monitoring, Observability and Analytics platform is presented in Figure 4.1. Each function will be explained in the following subsections.

4.2.1.1 The Infrastructure Data Analytics Function

The green block in Figure 4.1 represents the Infrastructure Data Analytics Function (IDAF) which is responsible for exposing infrastructure-related metrics.

Since the B5G NFs should be deployed in the form of either VMs or containers, herein, the infrastructure concerns both the bare-metal servers and the container orchestrator

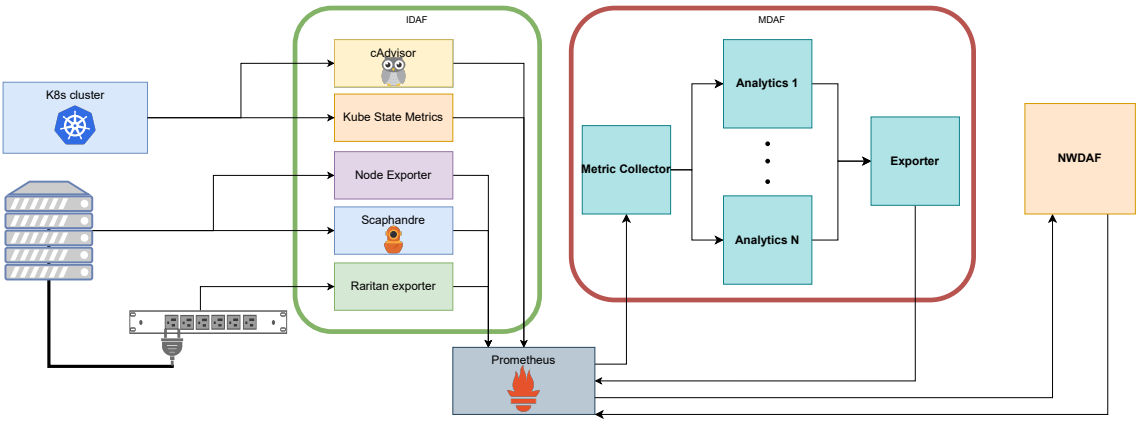

Figure 4.1: Structure of the whole Monitoring, Observability and Analytics platform ranging from the Infrastructure (IDAF) to the B5G Network domain (MDAF and NWDAF).

platform (e.g., Kubernetes in this case). In detail, I focus on containers rather than VMs, since the former is more flexible and "lighter" than the latter and currently preferred for deploying B5G NFs. However, it is worth highlighting from the start that the reasoning and also the development takes into consideration VMs as well.

Being the B5G network Provider and the infrastructure Provider usually different, it is essential to separate the data collection from the infrastructure and from the network. This reasoning leads to the separation between IDAF and MDAF. As a result, the MDAF will not have direct access to the infrastructure, but only to the metrics exposed by the IDAF. Additionally, this separation ensures that only the necessary data are available, and that the other, possibly sensitive, data is protected and not shared with the network.

As shown in Figure 4.1, the IDAF is composed of five components. All of them are open-source exporters compatible exposing metrics in a Prometheus-like format. cAdvisor [52] and KSM [53] expose information regarding the Kubernetes cluster. The former can be either embedded in kubelet (i.e., the main K8s agent responsible for managing the deployment of pods and for the health of the K8s cluster) or installed as a docker container and exposes the resource (e.g., CPU, memory, networking, etc.) utilization per container. The latter is an additional service that exposes the state of K8s objects (e.g., pods, nodes, replicaset, statefulset, etc.).

NodeExporter [57] and Scaphandre [55] expose metrics of the bare-metal servers. The former exposes a wide variety of hardware- and kernel-related metrics among which it worth mentioning the resource (e.g., CPU, memory, networking, etc.) utilization of the servers. The latter exposes the power consumption of every single process (including containers and VMs) running on the server. Scaphandre relies on the Intel Running Average Power Limit (RAPL) [36] counters and on the time spent by each process on the CPU to compute the power consumption per process. So far, the power consumption provided by Scaphandre includes only the CPU and, only in some cases, the DRAM controller consumption. Therefore, being the CPU the most power-hungry component of servers, the metrics should be somewhat reliable for processes not needing GPUs.

Finally, the Raritan exporter [56] is in charge of exposing the power consumption of the Raritan Power Distribution Units. In detail, I used the IX7™ PDU Controller by Raritan.

The aforementioned metrics are saved on a Prometheus [58] database which is common to all the three functions (i.e., IDAF, MDAF, NWDAF). Prometheus is a time series database collecting metrics composed of a timestamp, a value and optionally key-value labels.

4.2.1.2 The Management Data Analytics Function

The MDAF, originally defined in 3GPP TS 28.530 [75], is a NF belonging to the 5G SBA in charge of collecting and performing analytics on management KPIs for NFs or network slices.

Being the proposed MDAF herein power-efficiency focused, it is responsible for collecting power KPIs coming from all the cloud-native infrastructure resources (i.e., bare-metal servers, VM hypervisor, container management platform). As highlighted above, the MDAF does not have direct access to the infrastructure in order to keep separation among tenants; but can rather access only the metrics exposed by the IDAF.

Moreover, the MDAF may use AI/ML technologies to perform analysis or predictions on the data.

4.2.1.3 The Network Data Analytics Function

The NWDAF is responsible for the storage and data collection required for inference from the different NFs, as defined in the 3GPP TS 23.288. In particular, it collects additional analytics to allow inferring the MDAF preliminary mapping with control-level NF KPIs (e.g., the number and the type of PDU sessions, the distribution of UEs, the requests to each service exposed by NFs, etc.). It also computes KPIs through an analytics module and exposes the results to the NFs or external third-parties application. A prototype of NWDAF has been proposed by [77], which uses the Seasonal Auto-regressive Integrated Moving Average (SARIMA) model for data forecasting.

4.2.2 Description of the Proposed MDAF

The MDAF is devoted to collect and manipulate management data. Consequently, at its level, I suppose not to have knowledge regarding the network, but rather only regarding the infrastructure. As mentioned earlier, the proposed MDAF is power-efficiency focused, therefore the data mainly concern power consumption and resource utilization.

Due to virtualization, understanding the impact on power consumption of the single components of a 5G network is not trivial. In a nutshell, the MDAF obtains data from the IDAF and manipulates them to get the power consumption of the individual virtualized components (i.e., VMs and containers). As shown in Figure 4.1, the MDAF is essentially composed of three blocks: the metric collector, the analytics blocks (which may be more than one) and the exporter.

Let me start by providing a general overview of the MDAF routine that is shown in Algorithm 1. First, some configuration parameters are loaded: the URL of the Prometheus instance and the port on which the exporter is going to expose the new metrics. Then, an infinite loop consisting of collecting metrics and applying analytics begins. It is worth noting that the exporter is initialized and started only after the first round of collecting metrics and performing analytics because it should start only after the metrics to be exposed are already available.

Algorithm 1 MDAF routine

```
prom_config=load_prom_config()
prom_manager=PrometheusManager(prom_config)

exp_config=load_exp_config()
exporter_initialized=False

while True:

  scrape_metrics()

  apply_analytics()

  if exporter_initialized==False
    exp=mdaf_exporter (exp_config)
    exp.start_exporter(exp_config)
    exporter_initialized=True
```

Moreover, it is worth mentioning that all the code was written in Python and that the MDAF was actually run as a K8s container to be coherent with actual 5G deployment.

In the following subsections, details on the MDAF components are provided. Eventually, in Section 4.2.2.4, the added value of our MDAF is highlighted.

4.2.2.1 The Metric Collector

The metric collector exploits the prometheus-api-client [78] library to interrogate the Prometheus instance. The only configuration needed concerns the address of Prometheus: IP:PORT. Reminding once again that the proposed MDAF is power-efficiency focused, the metric collector is devoted to Scaphandre and cAdvisor metrics. Therefore, the main goal is to acquire the power consumption of each process and the resource utilization of each container.

It is of paramount importance to distinguish the processes based on types. Let me consider the set containing all the power per process metrics provided by Scaphandre: $S(t) = x_i(t)|i = 1, 2, \ldots, N(t)$ where $N(t)$ is the number of processes in execution at time t.

From now on, I remove the time dependency for simplification purposes. To distinguish among the types of processes, let me consider four non overlapping proper subsets of $S : K, D, V, E$ such that:

$$K \subset S, D \subset S, V \subset S, E \subset S$$

$$K \cup D \cup V \cup E = S$$

$$K \cap D = \varnothing, K \cap V = \varnothing, K \cap E = \varnothing, D \cap V = \varnothing, D \cap E = \varnothing, E \cap V = \varnothing$$

Then, each metric is assigned to one of the four subsets based on the following rule:

$$x_i = \begin{cases} x_i \in K | x_i & \text{is a K8s container} \\ x_i \in D | x_i & \text{is a Docker container} \\ x_i \in V | x_i & \text{is a virtual machine} \\ x_i \in E | x_i & \text{otherwise} \end{cases} \tag{4.1}$$

for $i = 1, 2, \ldots, N$. From an implementation point of view, the selection of the correct subset is based on analyzing the metric's labels.

Finally, it is worth deepening the role of subset E: it contains all the process' metrics that do not belong to virtualized components (i.e., containers (both K8s and Docker) and VMs). Therefore, subset E contains all the kernel-processes that are needed in order for the server, everything on top (hypervisor and the container orchestration platforms) to work correctly and for managing various operations such as reading/writing memory or handling packets. I assume that the servers would be off if not for the virtualized components (e.g., there may be a containerized 5G network running on K8s); hence the total power consumption of E represents the so-called "indirect" power consumption. The goal of the MDAF is be to assign the aforementioned power consumption to the virtualized components (i.e., to each element of the subsets K, D, V) in a balanced way.

4.2.2.2 Analytics

The Analytic block(s) are where the "intelligence" resides: its structure allows to plug in different algorithms depending on the current needs. In principle, herein any kind (i.e., possibly involving Artificial Intelligence as well) of data manipulation can be performed. These algorithms should be run online (i.e., while the B5G network is up and running and serving UEs) in order to both provide manipulated data upwards (e.g., to the final user) and to apply optimization decisions on the network (e.g., migration, scaling) in a live fashion.

In the proposed MDAF, only few simple operations are performed in order to finally compute the "indirect" power consumption of the virtualized components.

First, however, it is worth mentioning that an easier task concerning metric enrichment is performed. This is necessary to cope with a Scaphandre issue: it is not able to acquire labels for containers using containerd as Container Runtime Interface (CRI). This, therefore, affects K8s containers which can still be uniquely identified through the container id, but do not have essential labels such as pod name and namespace. To solve this, I propose to merge the data coming from KSM with the one from Scaphandre.

Then, focusing on the core purpose of the proposed MDAF, in the following I explain the algorithms that are used to map the kernel-level power metrics mentioned in the previous subsection. First of all, I divide the $x_i \in E$ into categories. Such association is done "manually" thanks to a JSON file containing the associations between execution names and categories. This method was chosen to have a precise categorization. The chosen categories are 19 and are shown in Figure 4.2. While the JSON file can and was updated after a thorough testing, another process classification mechanism could be considered to adapt the classification to a wide range of heterogeneous environments (i.e., featuring different kernel and background processes) and to the updates which any software application is subject to.

Once all the kernel processes are categorized, the total power consumption per category can be mapped onto the actual components of interest (i.e., containers and VMs).

Herein I focus on containers (both K8s and Docker) as the essential ones for the following reasons. First, because this way I can leverage on the resource utilization metrics provided by cAdvisor. Secondly, because containers are preferred when implementing a B5G network because they are lightweight (compared to VMs). Even though most B5G NFs are implemented on K8s since it offers a layer of orchestration (compared to Docker) and therefore makes instantiation and reconfiguration operations easier; I also keep a tight

support for Docker containers since they are usually preferred for the containers needing multiple interfaces (i.e., the UPF).

The "indirect" power consumption is mapped differently according to the category as shown in Figure 4.2. Most categories, the ones framed by the red, pink and green rectangles, are mapped to containers according to their percentage of resource utilization. An example of such algorithm for the CPU utilization is shown in Algorithm 2. This algorithm allows to ascribe more power to the most resource-hungry containers. As far as resources go, three are taken into account: CPU, memory and networking. The network and network management categories are associated to networking utilization (in particular to number of transmitted bytes), while the memory management one is associated to the memory utilization. The remaining categories are associated, in lack of an obvious link, to the CPU consumption, since said component is the most power-hungry in absence of

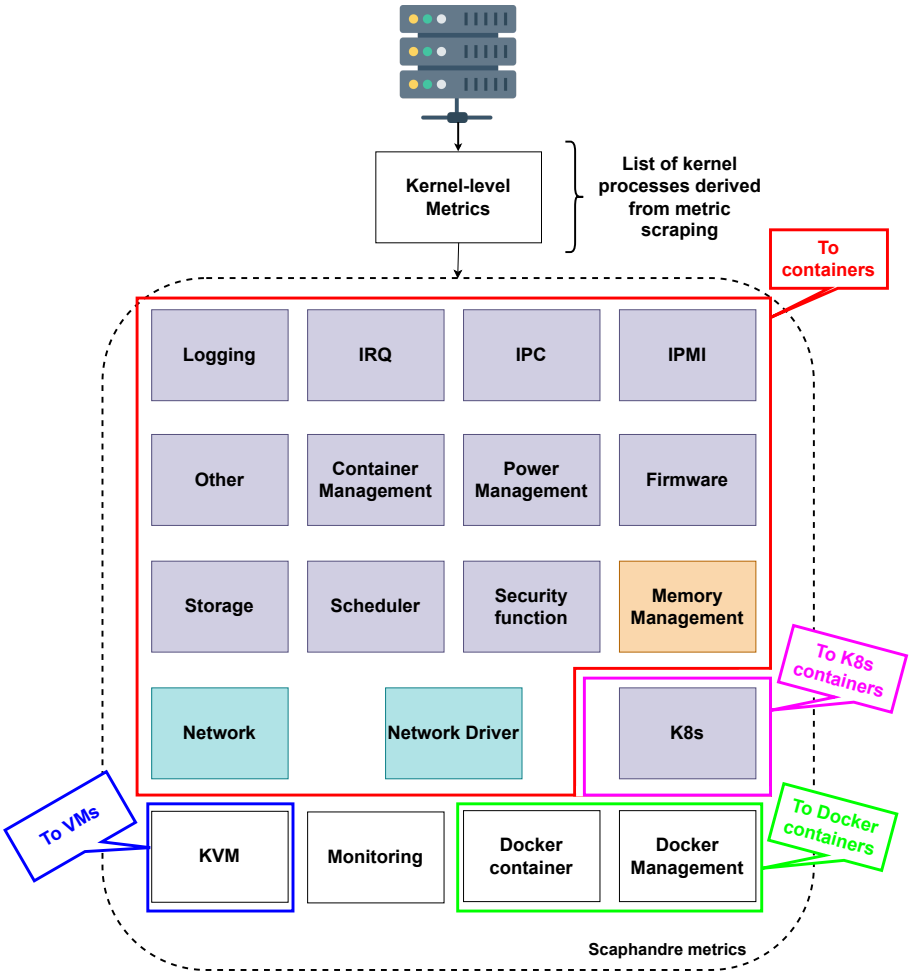

Figure 4.2: Categorization of the kernel-level metrics (i.e., $\forall x_i \in E$)

Graphics Processing Units (GPUs). Then, the categories framed by the pink and green rectangles are mapped according to the CPU consumption to only the K8s or only to the Docker containers. It is worth mentioning that the processes boxed in the "Other" category are the ones which are difficult to relate to an operation/resource: for instance, the command line commands (e.g., sudo, grep, etc...) and the kworkers. The kworkers are processes that perform most of the actual processing for the kernel, especially in cases where there are interrupts, timers, I/O, etc. They may contain hints about their tasks in their execution names, however, it is generally difficult to gain insights about them.

The VM-related category (framed by the blue rectangle), in absence of resource utilization, is mapped in a uniform way to the VMs as shown in Algorithm 3.

Finally, the monitoring category is not mapped to any virtualized components, since it contains the power of monitoring processes that were not deployed as containers.

4.2.2.3 The Exporter

The goal of the exporter is to expose the newly generated metrics in a Prometheus-like format. It leverages once again on the prometheus-api-client library to respect the Prometheus format. Moreover, the port on which it exposes the metrics is configurable.

An exhaustive list of the aforementioned metrics is provided in Table 4.1. Finally, it is worth mentioning that all of them are of Gauge type, that is a variable that can either decrease or increase, since they are representing power.

Algorithm 2 Mapping of kernel-level power according to resource utilization

r_i= Get CPU utilization per container from *cAdvisor*
total_cpu_utilization=0

$\forall i \in r_1$:
    total_cpu_utilization+=r_i

$\forall x_i \in K$:
    % = r_i/total_cpu_utilization
    x_i+ = %· category_power_cons
$\forall x_i \in D$:
    % = r_i/total_cpu_utilization
    x_i+ = %· category_power_cons

Algorithm 3 Mapping of kernel-level power proportional to the number of VMs

$\forall x_i \in V$:
  % = 1/#_vms
  x_i+ = %· category_power_cons

Metric name	Type	Labels	Description
mdaf_enriched_k8s_ container_power_microwatts	Gauge	container_id, node, namespace, pod, cmdline, exe, pid	MDAF power consumption of K8s containers the additional labels (pod and namespace) which are missing from scaphandre (to cope with Scaphandre issues with containerd)
mdaf_aggregated_k8s_ container_power_microwatts	Gauge	container_id, node, namespace, pod, cmdline, exe, pid	MDAF power (+ indirect) consumption of K8s containers
mdaf_aggregated_docker_ container_power_microwatts	Gauge	container_id, container_names, container_scheduler, node, cmdline, exe, pid	MDAF power (+ indirect) consumption of docker containers
mdaf_aggregated_vm_ power_microwatts	Gauge	vmname, node, cmdline, exe, pid	MDAF power (+ indirect) consumption of VMs
mdaf_category_ power_microwatts	Gauge	node, category	MDAF power consumption of each category (composed of kernel-level metrics)
mdaf_monitoring_ power_microwatts	Gauge	node	MDAF power consumption of the monitoring category (excluding docker containers). This metric will be empty unless you install some monitoring components directly as a service (e.g., Scaphandre)

Table 4.1: Metrics exposed to Prometheus by the MDAF.

4.2.2.4 The Added Value of the Proposed MDAF

The goal of this section is to highlight the added value of the proposed MDAF with respect to standard energy/power monitoring solutions (e.g., Scaphandre). In further detail, Scaphandre considers only the CPU power consumption (provided by Intel RAPL) and divides it, based on the CPU utilization, onto all the server processes. Thus, Scaphandre takes into account only the CPU utilization. Furthermore, it doesn't ascribe the "indirect" costs to VMs and containers. Clearly, considering only the CPU power consumption makes Scaphandre a good tool of comparison only in the case of CPU-intensive containers/VMs. While this is suitable for B5G NFs, it would be a great limitation in the case of GPU-intensive containers/VMs in which scenario another tool of comparison should be found.

Computing the power consumption of a single VM or a single container is not trivial at all. The goal is to provide a comprehensive power consumption per container/VM that considers the "indirect" power consumption as well. Here lies the difference between the proposed MDAF and Scaphandre. My rationale is based on the assumption that, in the absence of NFs, the servers hosting them would be turned off and consequently would be

consuming zero power. Therefore, the goal of the proposed MDAF is to map all the power consumption of the servers to the NFs (i.e., VMs or containers).

Therefore, the "indirect" power consumption is mapped only to the processes of interest (i.e., mainly containers, but also VMs). In order to also take into account the other physical components utilization (not only CPU, but also memory and networking), the mapping is carried out, as explained in Section 4.2.2.2, through the CPU, memory, and networking utilization of the containers. Consequently, for instance, to the containers using more networking resources (sending and receiving more bytes) will be ascribed a bigger portion of the kernel-processes involving the networking area.

In a nutshell, the proposed MDAF provides a comprehensive power consumption of each NF; this is the first step to measure the power consumption of slices and end-users/vertical applications. Consequently, this begins the effort to address the need to spread awareness of the power consumption (which is, by definition, actually consumed at the infrastructure level) to all the stakeholders involved in the B5G ecosystem (from infrastructure providers up to MNOs and end-users/verticals).

4.3 Experimental validation

4.3.1 Experimental Setup

The aim of the experiments reported in this section is to conduct measurements on a 5G network to show the usefulness of the proposed MDAF. In order to have a completed environment composed of both the core and the access network, I used an open-source 5G network Open Air Interface (OAI) [79] for the former, and a commercial product (i.e., LoadCore, [80]) for the latter. LoadCore is a 5G network simulator that makes use of "agents" to simulate a whole 5G network or only some components.

In further detail, the setup consists of two physical servers: one acts as the K8s master and the other one as the K8s worker. The master hosts most of the monitoring applications mentioned in Section 4.2.1.1 and all the containers needed to enhance the capabilities of Kubernetes (e.g., LoadBalancer, etc.). The K8s worker hosts the OAI 5G network under the form of K8s containers except for the UPF which is deployed as a docker container, the monitoring application needed running on both servers (i.e., Scaphandre and NodeExporter), the LoadCore components and the MDAF that, as mentioned earlier, is deployed as a K8s container since it would be part of the Control-Plane of the 5G network. Details of the full 5G network deployment are shown in Figure 4.3. The white-coloured

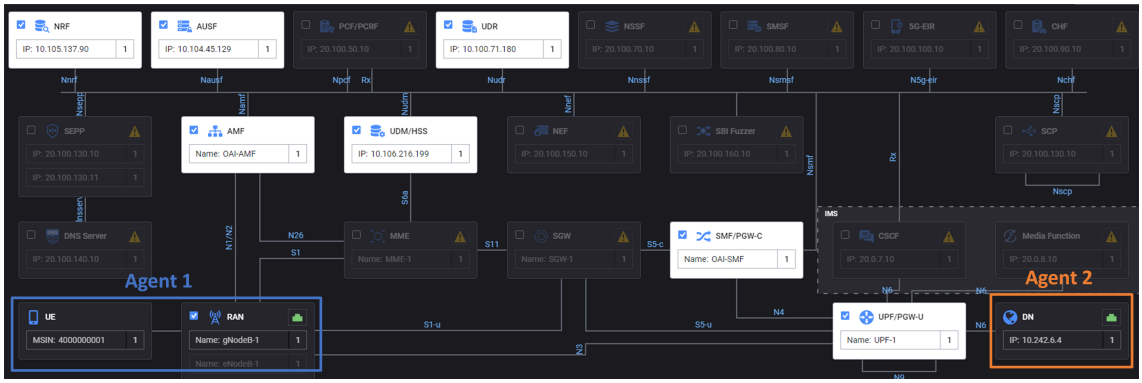

Figure 4.3: Details of the test setup as shown by the LoadCore dashboard. The NFs in the white boxes are provided by OAI, while the components encased in the blue and orange rectangle are simulated by LoadCore respectively by two different agents.

NFs are the ones provided externally (herein by OAI), the RAN (composed of gNodeB and UEs) and the DN are simulated by LoadCore respectively by two agents deployed as VMs.

During the tests, both User- and Control-Plane operations were investigated. For both cases, data were collected from both Prometheus and LoadCore with a 3-second sampling period in the form of csv files.

For the User-Plane, UE(s) generating stateless UDP data flows were investigated. In further detail, independent UDP flows are generated from the UE(s) to the data network and vice versa. In this case, the offered load and the number of UEs were changed. First, I tested 1 UE generating a varying load from 1 kbps up to 100 Mbps (i.e., the maximum supported by OAI in this configuration). Then, I tested changing the number of UEs (1, 10 and 100) while keeping the aggregate offered load constant.

Regarding the Control-Plane, I tested two simple operations: UE attachment and PDU session establishment. In the former, I examined the situation in which 100 UEs are attaching with a 1 UE per second rate (i.e., the maximum allowed). In the latter, a single UE established PDU sessions with a rate of 1 per second; herein I have found the maximum number of PDU sessions handled by OAI to be equal to 85.

Finally, it is worth mentioning that each test was performed 10 times. Since no significant difference was noticed among the instances of the tests and since the temporal analysis has the most use, in the following only one instance for each test is presented.

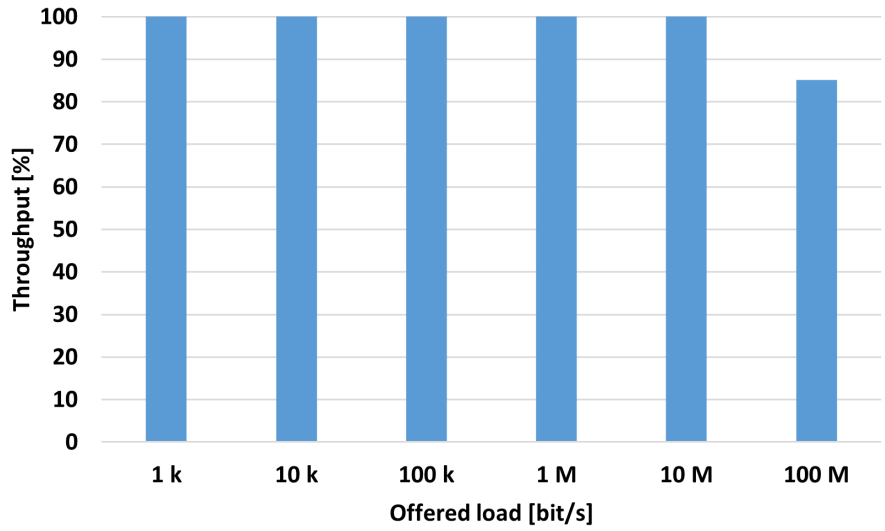

Figure 4.4: Throughput obtained for the UPF under tests at varying offered load.

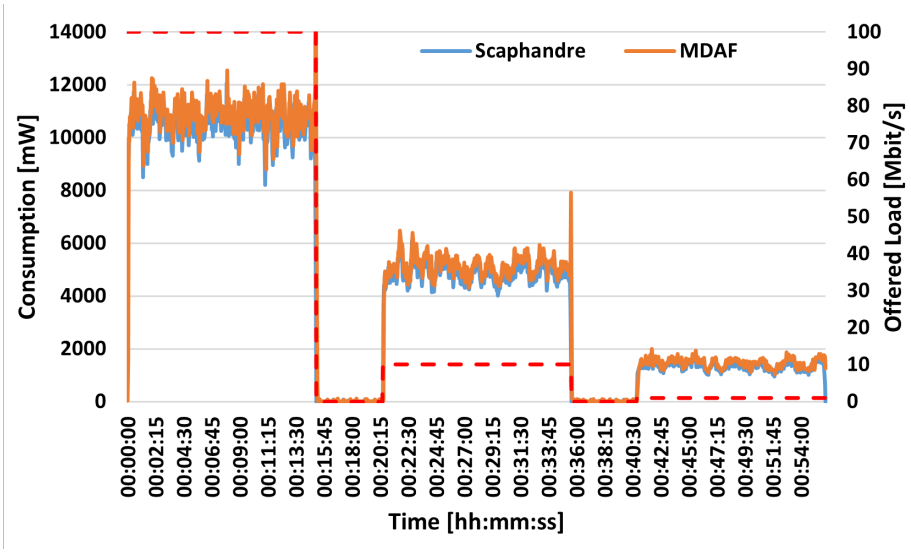

Figure 4.5: Power consumption of the UPF measured for Scaphandre and MDAF during the test (part 1).

4.3.2 Experimental Results

4.3.3 User-Plane

The first tests involved User-Plane traffic, as explained in the previous section, and allowed assessing the accuracy of the proposed MDAF prototype with respect to the state-of-the-art solution Scaphandre.

They involved measuring the power consumed by the UPF when User-Plane traffic is generated by one UE at varying rates, decreasing from 100 Mbit/s to 1 kbit/s. For each rate, traffic was transmitted for 15 minutes followed by 5 minutes of idle.

Figure 4.4 reports the throughput obtained for the different offered loads. It can be seen that the OAI UPF is able to reach 100% throughput for traffic below 100 Mbit/s but experiences losses when the latter load is transmitted. This fact is only due to OAI

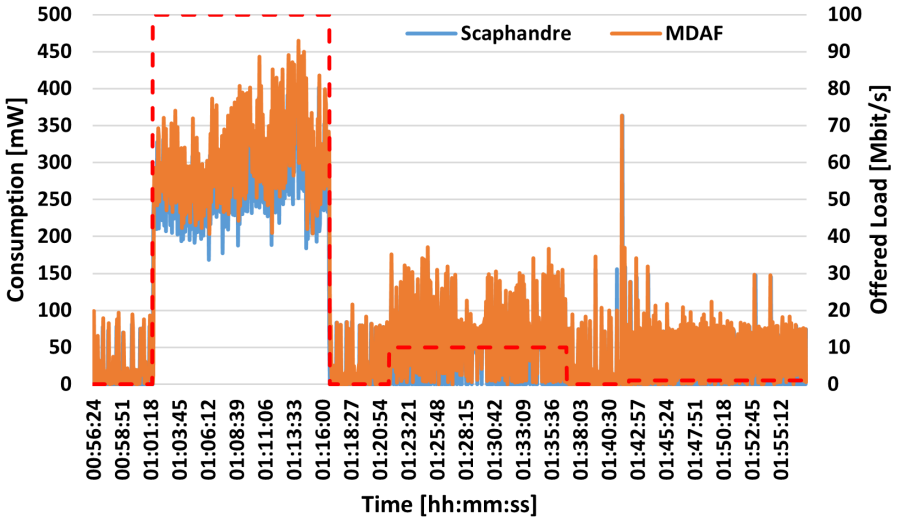

Figure 4.6: Power consumption of the UPF measured for Scaphandre and MDAF during the test (part 2).

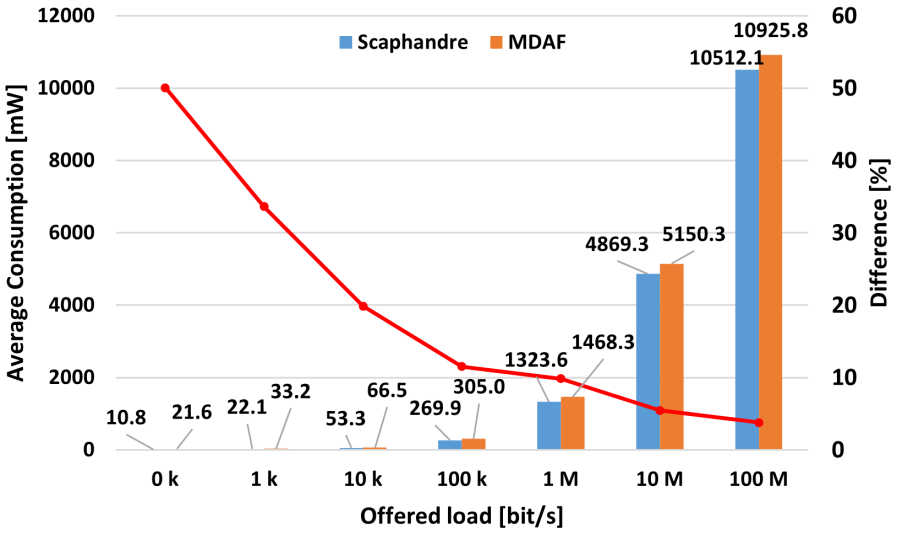

Figure 4.7: Average consumption obtained for the UPF under tests at varying offered load.

limitations and has no implications on my evaluation, but is worth reporting nevertheless for a coherent understanding of the following results.

Figures 4.5 and 4.6 show the power consumption, at varying loads (represented with the red dotted line plotted on the secondary y axis), for both Scaphandre and MDAF. Although the general trend is similar, the graphs show a slightly lower consumption for the former. The difference between Scaphandre and MDAF is more evident by looking at Figure 4.7, which reports the average power consumption for the different traffic loads and the percentage of the difference between the two tools, which is 50% in the absence of traffic, 33% for 1 kbit/s and going down as traffic grows to end slightly below 4% for the highest offered load.

Although the consumption not detected by Scaphandre is not very high (the average difference for 100 Mbit/s is around 400 mW), when this underestimation occurs for all the functions deployed in a network, the impact can be severely critical and lead to misinformed resource allocations. This is especially true for the UPF that, being the sole User-Plane NF, is usually present in multiple instances in the network.

Reminding that the difference in the consumption measured by Scaphandre and MDAF is due to the "indirect" contribution of the kernel-level power metrics, it is possible to gain a better understanding of the data reported in Figures 4.5 and 4.6 by taking a look into such metrics. As explained in Section 4.2.2.2, the consumption arising from the (categorized) kernel processes is assigned to a specific component according to its resource utilization (e.g., CPU, network and memory), a piece of information provided by the cAdvisor component in the IDAF. Figure 4.8 reports the breakdown of the contribution to the UPF provided by the most power-hungry categories of processes. It is worth noting that all these categories are assigned according to the CPU utilization, which means that processes ascribed upon memory and network do not cause such a relevant overhead to the overall consumption of the UPF.

By looking again at Figure 4.5, a spike in the consumption measured by the MDAF can be noticed when traffic transmission ends, which is particularly visible at 00:15:45 and 00:36:00, but is not detected by Scaphandre. By investigating the consumption ascribable to each category individually, it can be noticed that the spikes are mainly caused by the processes belonging to the "Container Management" and "Scheduler" categories, which are the highest in Figure 4.8. Figure 4.9 reports their consumption obtained during the same test case as in Figure 4.5, and one can see how their individual spikes account for 3500 and 1000 mW, respectively, which explains the 4000 mW spike from Figure 4.5. The same consideration can be made for the second spike as well. Regardless of the reason

for that surge in consumption, the proposed MDAF not only allows to detect it, but it also provides an outline of the processes involved to give the chance for proper investigation and identification of potential issues in the deployed instances.

An additional test was also carried out to measure the power consumed by the UPF in the presence of more than one UE. Namely, I tested two different loads, 1 Mbit/s and 100 Mbit/s, but this time the load was obtained by aggregating the traffic generated by 1, 10 and 100 UEs.

Figures 4.10 and 4.11 report the throughput and the power consumption obtained at varying traffic and number of UEs generating it. In the User-Plane, it can be seen that no differences appear when transmitting traffic from one or more than one UE so, for the sake of brevity, for the remaining of the paper we only considered the case of one UE. On the contrary, the higher number of UEs causes an overhead on the Control-Plane due to UE registrations and PDU session establishments. However, this is not reported because, as shown in the following Section, the power consumption of the Control-Plane is much lower in amplitude and lasts for a very short time compared to the User-Plane; as a result, its contribution is negligible.

Finally, it is worth discussing what would happen to the User-Plane results in different scenarios. First of all, the type of traffic can be analysed. Herein, only UDP was chosen since it is the simplest type of traffic and it can model well some B5G applications (i.e., video streaming). Another possible choice could have been TCP: a reliable transport

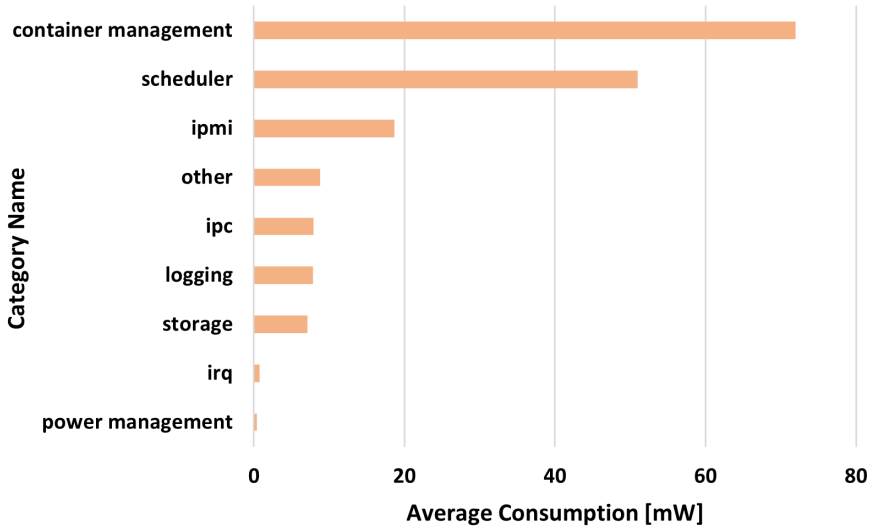

Figure 4.8: Breakdown of the consumption ascribable to the UPF for the most relevant categories.

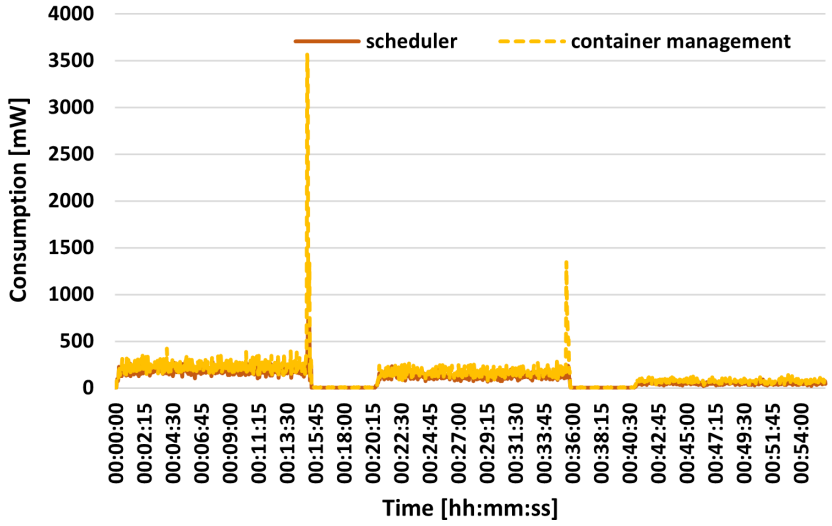

Figure 4.9: Consumption of the "container management" and "scheduler" categories ascribed to the UPF.

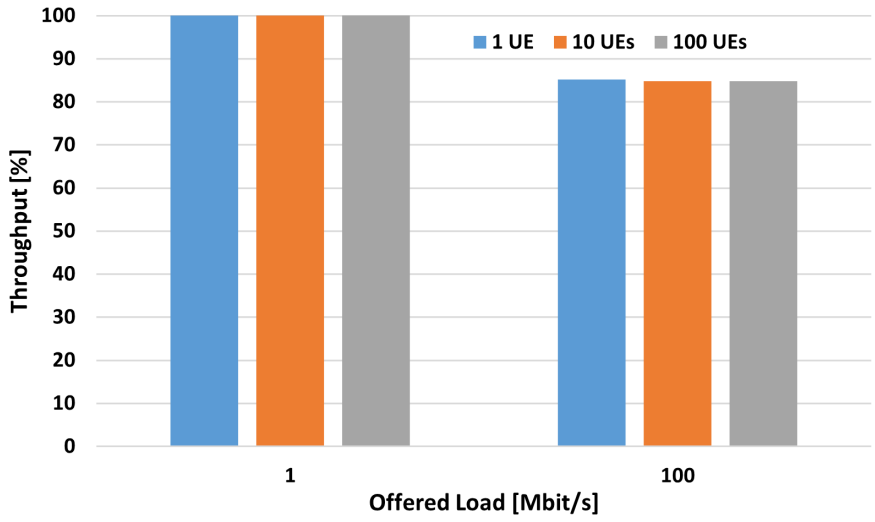

Figure 4.10: Throughput obtained for the UPF under test at varying offered load and number of UEs.

protocol that, through acknowledgments, retransmits lost packets. In this case, a difference would be detected in the highest load case (100 Mbit/s): the lost packets would be retransmitted; therefore, the power consumption and the throughput would increase. In case of a mixed type of traffic (i.e., UDP and TCP), all of the above would still hold. Another change in scenario can involve the regularity of the traffic. For instance, in order to model a real B5G network traffic, the offered load would need to change frequently

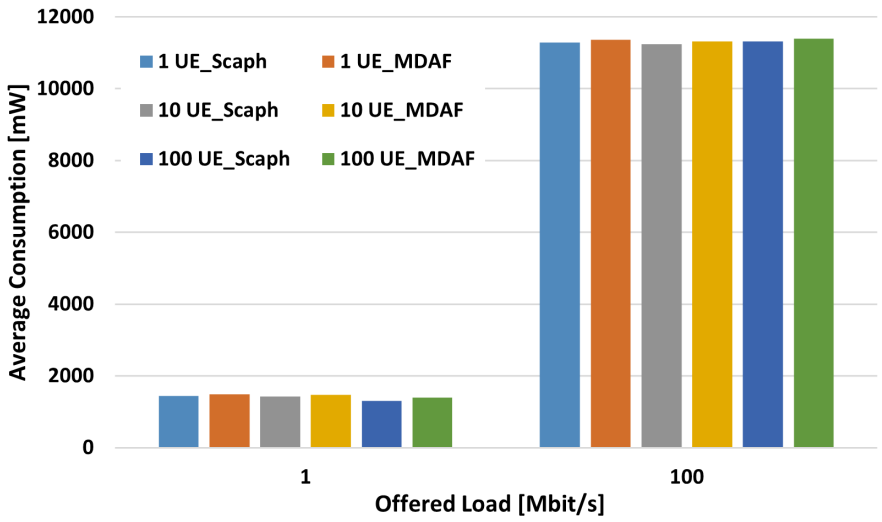

Figure 4.11: Average consumption obtained for the UPF under test at varying offered load and number of UEs.

during the day and it would have a peak during the most busy hours while having a fall during the least busy ones. The irregular traffic load would result in several changes in the resource utilization of the UPF. This would possibly change the assignment of the physical resources to the UPF container and, consequently, the overall power would differ due to the variation in the "sleep" states to which the CPUs are assigned.

4.3.4 Control-Plane

For the evaluation of the consumption ascribable to the Control-Plane, we chose two operations, namely UE attachment and PDU session establishment, and measured the consumption of several NFs involved in these operations with the two tools. Namely, the AMF, AUSF, SMF and UDM have been investigated.

The selection has been made because these functions perform a plethora of operations that occur upon diverse triggers: the AMF is responsible for all the signalling related to user authentication and mobility, including non-access stratum protocols; the AUSF further provides the support for the authentication of untrusted non-3GPP access; the SMF is responsible for the lifecycle management of PDU sessions; finally, the UDM is a database that stores runtime information used by the AMF. For both UE attachment and PDU creation, tests lasted around nine minutes and we collected the power consumed by each considered NF.

In the first test, LoadCore was used to simulate the attachment and de-attachment of 100 UEs, one at a second. Figures 4.12-4.15 show the consumption for the AMF, AUSF, SMF, and UDM, respectively, measured with Scaphandre and MDAF, along with the number of attached UEs, reported with the dotted line.

For all of these functions, one can recognize the same trend happening for the UPF, albeit at a smaller scale: while Scaphandre allows to detect the changes due to the attachment and detachment of the UEs, which is reflected by an increase of the power consumption at 3:00 and a decrease at 4:45 when the UEs begin detaching, the inability to account for the "indirect" consumption results in an underestimation that on average varies, depending on the NF, between 5 and 14 mW, as reported in Figure 4.16. In this figure, it can be noticed that, while AMF and SMF overall consume less, the consumption detected with the MDAF is more than double the amount reported by Scaphandre. Moreover, it can be noted that, in the absence of attached UEs, Scaphandre only detects some repetitive consumptions that correspond to the "keep-alive heartbeats" towards the NRF.

The second test considered the creation of a growing number of PDU sessions up to 85, which is the current limit for OAI, their maintenance for 36 seconds, and their release. Figures 4.17-4.20 show the consumption of the same NFs as in the previous case, and similar considerations can be drawn. Here, the main difference is that AUSF and UDM do not show any significant variations when PDU creation/destruction operations occur. This can be easily explained by the fact that the AUSF and UDM are involved in

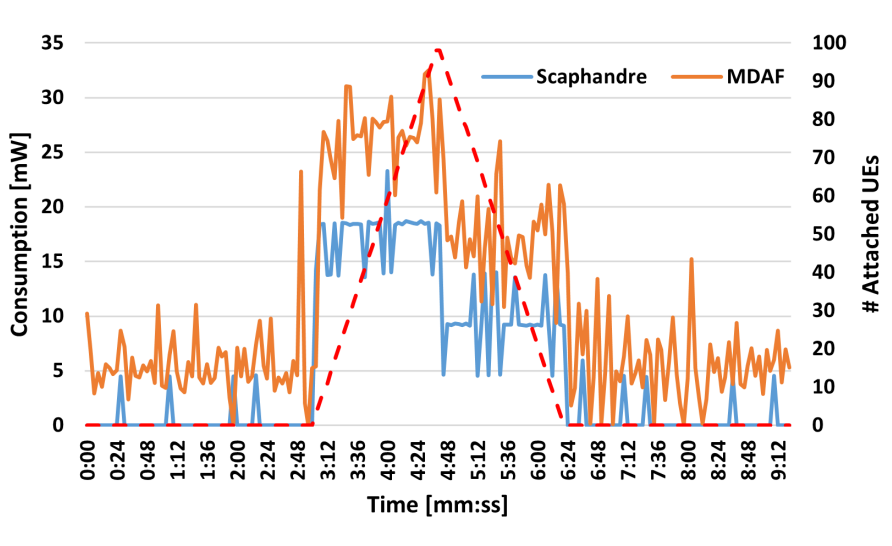

Figure 4.12: Power consumption of the AMF measured for Scaphandre and MDAF during UE attachment operations.

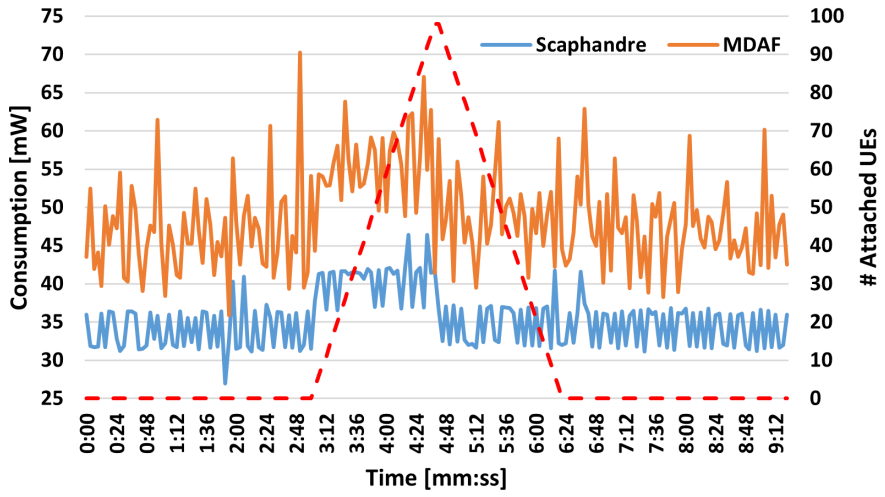

Figure 4.13: Power consumption of the AUSF measured for Scaphandre and MDAF during UE attachment operations.

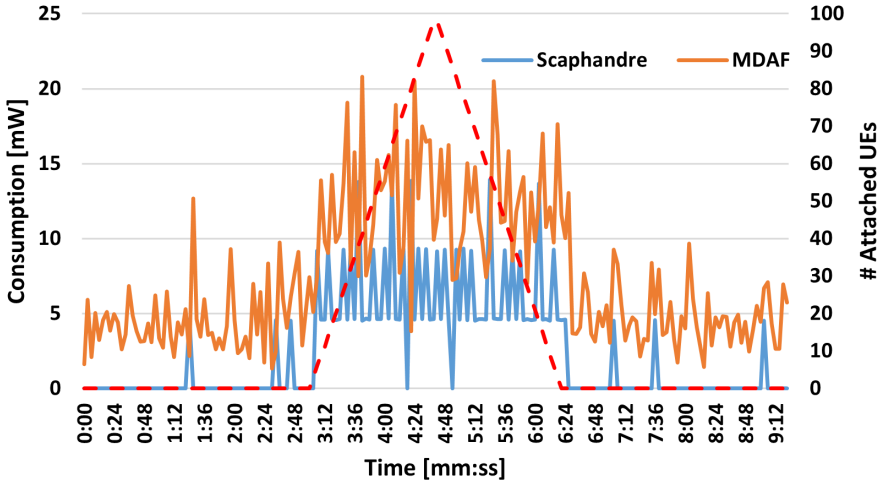

Figure 4.14: Power consumption of the SMF measured for Scaphandre and MDAF during UE attachment operations.

the UE authentication when it attaches to the network; therefore they are “stressed” only at the beginning of the PDU session establishment test. Moreover, it can be seen that the consumption measured by Scaphandre for these two NFs is not only around 13 mW lower with respect to the MDAF (see Figure 4.21), but it also presents less variations. Focusing on the AMF and SMF, the same behavior in the absence of operations on the PDU sessions as in the previous test appears; additionally, the difference between

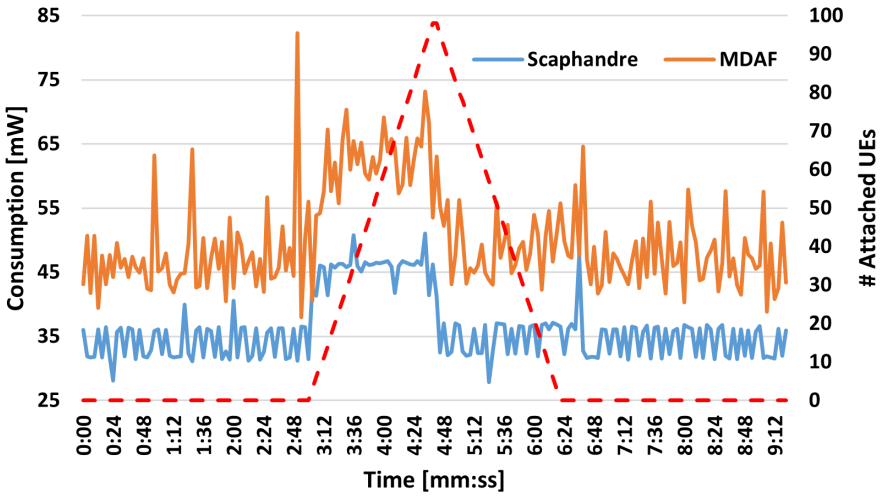

Figure 4.15: Power consumption of the UDM measured for Scaphandre and MDAF during UE attachment operations.

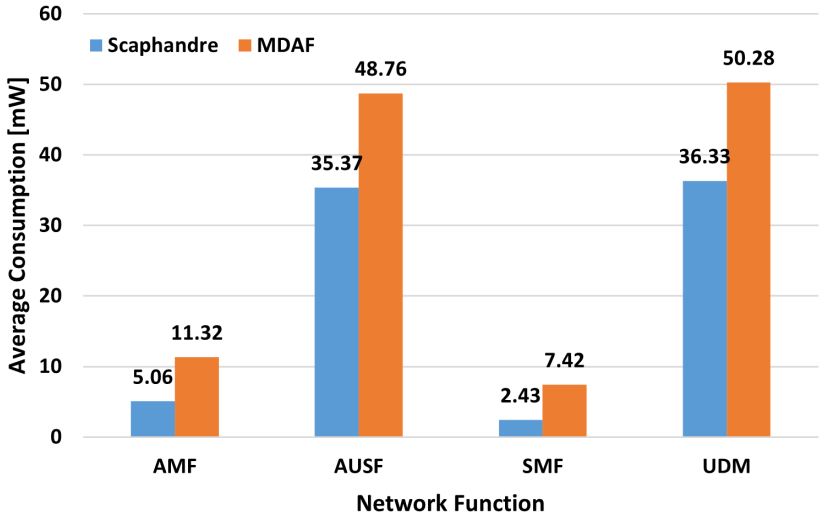

Figure 4.16: Average consumption obtained for the Control-Plane functions during UE attachment operations.

the consumptions measured with Scaphandre and MDAF grows, emphasizing a stronger impact on CPU operations that Scaphandre alone is not able to detect.

4.4 Conclusions

In recent years, mobile networks have focused on targeting the goal of energy efficient. Consequently, the need for accurate power/energy measuring tools has now become apparent.

In this chapter I presented a power-focused MDAF which is a crucial block of the B5G Observability framework. The MDAF is the NF that can bridge the gap between the infrastructure and the network layer which are usually owned by different entities. It can infer to upper layers (network and end-users/verticals) management information coming from the infrastructure. The MDAF we propose herein has a modular structure that allows to add other analytics modules (potentially exploiting AI) further down the line.

Tests were performed on both the User-Plane and the Control-Plane of the network, and the MDAF accuracy was compared with a common power monitoring tool: Scaphandre.

Overall, results show that the MDAF measures a higher power consumption than Scaphandre. This is due to fact that I take into account also the "indirect" power consumption of each virtualized entities (i.e., containers and VMs). The difference ranges from a minimum of 4% (with a 100Mbit/s offered load) to a maximum of 50% (when no traffic is generated). It is also worth mentioning that the MDAF is able to detect peaks that Scaphandre does not. Results show that the "indirect" power consumption is mainly

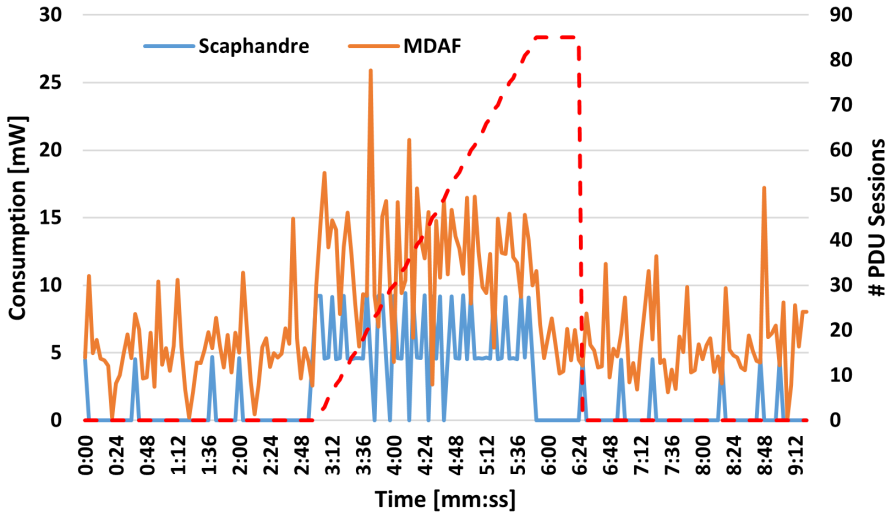

Figure 4.17: Power consumption of the AMF measured for Scaphandre and MDAF during during PDU session lifecycle operations.

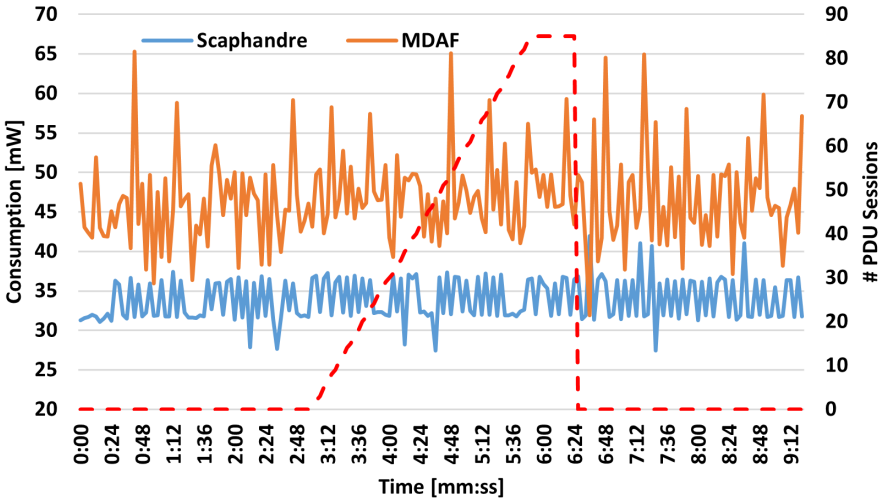

Figure 4.18: Power consumption of the AUSF measured for Scaphandre and MDAF during during PDU session lifecycle operations.

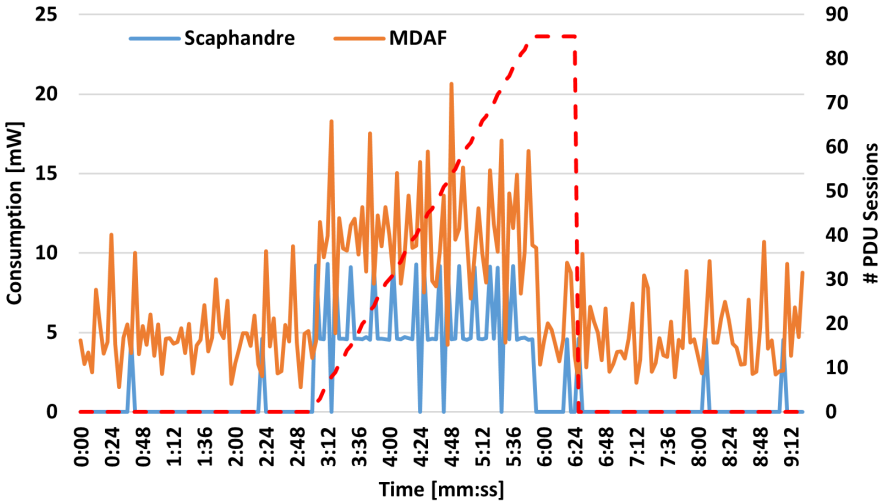

Figure 4.19: Power consumption of the SMF measured for Scaphandre and MDAF during during PDU session lifecycle operations.

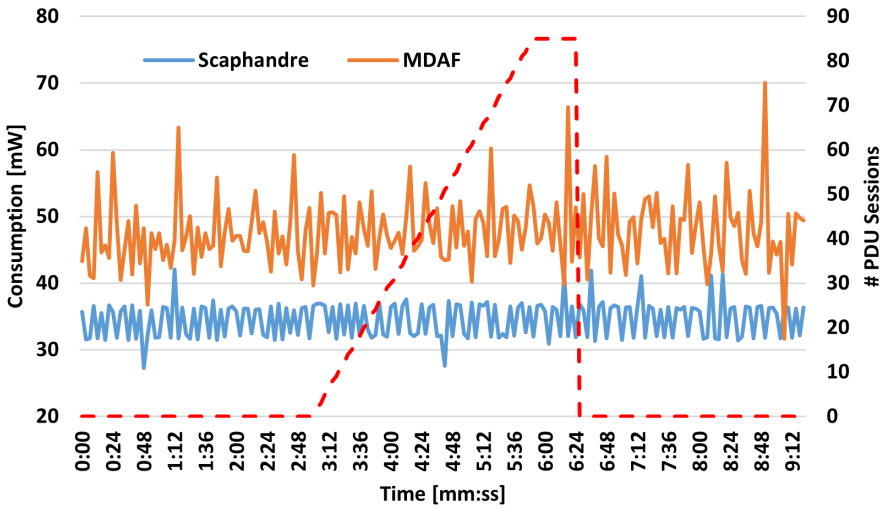

Figure 4.20: Power consumption of the UDM measured for Scaphandre and MDAF during during PDU session lifecycle operations.

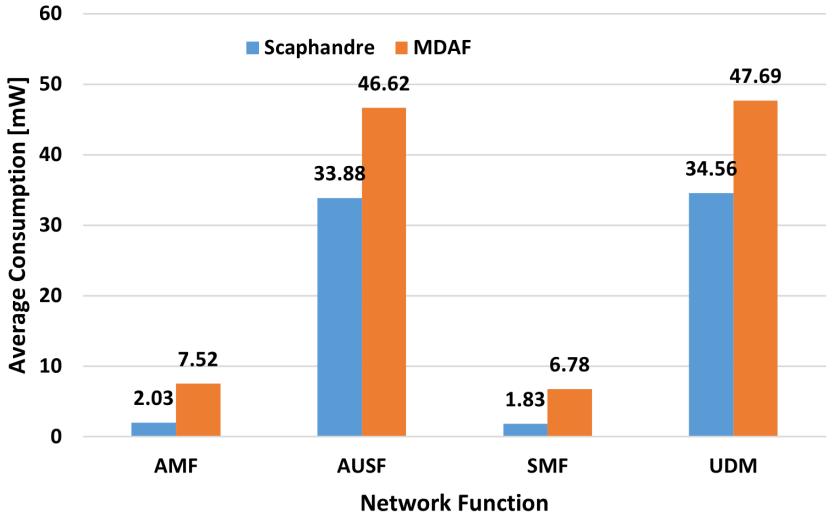

Figure 4.21: Average consumption obtained for the CP functions during PDU session lifecycle operations.

ascribable to the kernel-processes related to “Container Management” and “Scheduler” operations.

The additional power may not seem too high in the results shown here, however it is crucial to highlight that, in real conditions, the NFs would have multiple instances (this is especially true for the UPF) and therefore the impact of the “indirect” power would, in fact, be much higher.

The proposed MDAF generates more accurate power measurements per NF. This is the first step required to spread an energy/power awareness to upper stakeholders. Thus, while the MDAF provides information at the single NF level of granularity; the future works include the level just above: the network slice. The measurements presented in this chapter can be aggregated in order to estimate the power consumption of each B5G network slice.

Acknowledgements

This work has been partially supported by the Horizon Europe Research and Innovation Action 6Green (Grant Agreement no. 101096925) and by the European Union - Next Generation EU under the Italian National Recovery and Resilience Plan (NRRP), Mission 4, Component 2, Investment 1.3, CUP F83C22001690001, partnership on "Telecommunications of the Future" (PE00000001 - program "RESTART") in the Net4Future project.

CHAPTER 5

Low Power Idle in B5G Networks

5.1 Introduction

After exploring throughly the issue of power measurement in B5G networks in the previous two chapters; in this chapter, the flexibility of B5G NF implementations (i.e., VMs or containers) is exploited to present a power saving analysis based on one of the most common general purpose power saving technique: LPI (see Section 2.3 for details).

This study tackles one of the issue that has arisen from the lack of a common framework [41] between the owner of the physical infrastructure and of the applications and network services running on top of said infrastructure. This has prevented the proper application of the power management schemes natively available in the servers, causing a waste of resources and, consequently, an increase in their carbon footprint and energy requirements.

Herein, I examine the tradeoff between power consumption and performance, under different power management configurations, of the most power-hungry NF of B5G networks: the UPF. UPF is the only data plane function, and is responsible for the routing and forwarding of data plane packets towards the correct DN (in uplink) or the correct gNodeB (in downlink).

Different UPF implementations exist. It is worth mentioning that they can be deployed either on general-purpose servers or on hardware acceleration devices (e.g., P4 switches). In the following, I consider the former and I examine the UPF power consumption and performance while applying power management techniques on such general-purpose servers.

Among the most common power management techniques, the LPI is taken into consideration. It forces CPU cores to enter "sleeping" states when not processing any task. The impact of such power management capability has already been presented in a number

The content of this chapter has been published in the following paper:

R. Bolla, R. Bruschi, D. Cassinelli, F. Davoli, C. Lombardo, N. S. Martinelli and B. Siccardi, "Performance and Power Analysis of a 5G UPF Varying the C-States of the Hosting Server," 2025 IEEE International Conference on Communications Workshops (ICC Workshops) (2025): 1025-1030.

of studies [81][82]. Thus, the goal of this Chapter is to study the power consumption and performance of a B5G UPF, while acting on the "sleeping" states of the hosting server and find the best tradeoff between power saving and performance.

5.2 Methodology

5.2.1 The Advanced Configuration Power Interface Specification

In this Chapter I exploit the LPI mechanism in order not to decrease the performance (which would correspond in high delays or packet loss in the B5G network in this case) and to take advantage of the burstiness and seasonality of the mobile traffic through the "sleeping" states. In the following, details about the server hosting the UPF are provided. The hosting server on which the UPF was tested has an Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz. In particular, this CPU model has 56 logical CPUs which can assume the following 5 C_x states:

1. C_0: it corresponds to the active state in which the CPU can execute tasks.
2. C_1: it corresponds to the "lightest" sleeping state. In this state the Core clock is off. The processor is not executing instructions but can return to an executing state almost instantaneously.

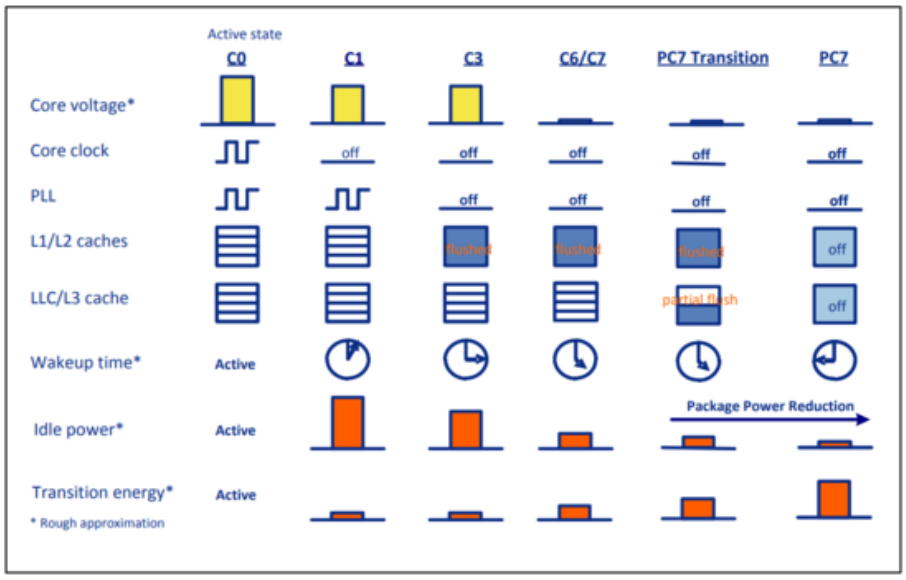

Figure 5.1: C_x states and their effects [8].

3. C_{1E}: it corresponds to an enhanced C_1 state. Herein the latency to return to C_0 is slightly increased, while the power slightly descreases.

4. C_3: in this state, clocks are stopped and the local caches (L1/L2) of the core is flushed, and the core is powered down.

5. C_6: it corresponds to the "deepest" sleep state. Herein, in addition to all the actions taken in the previous sleeping states, the shared cache (L3/LLC) of the package is flushed and at the end the package/whole CPU can be (almost) powered down.

A summary of the effects of the above C_x states is depicted in Figure 5.1.

5.2.2 The User Plane Function

The User-Plane is only composed of the UPF. This is usually the most power-hungry NF in 5G networks since it is the one responsible for the actual routing and forwarding of data packets. As shown in Figure 5.2, the UPF has three interfaces: N4 towards the SMF which is used to configure the PDU sessions and therefore involves control plane traffic, and N6 and N3 interfaces towards the DN and the gNodeB, respectively, where the data plane traffic is sent.

Usually, being UPFs VMs or containers, they run on general-purpose servers. However, it is worth mentioning that different implementations (either relying on hardware or software acceleration devices/techniques) of UPF exist. In the literature, studies can be found comparing such implementations from a performance [83] point of view. The simplest UPF implementation corresponds to a user-space application. In order to increase the performance, accelerated UPFs exist; they exploit either software or hardware acceleration

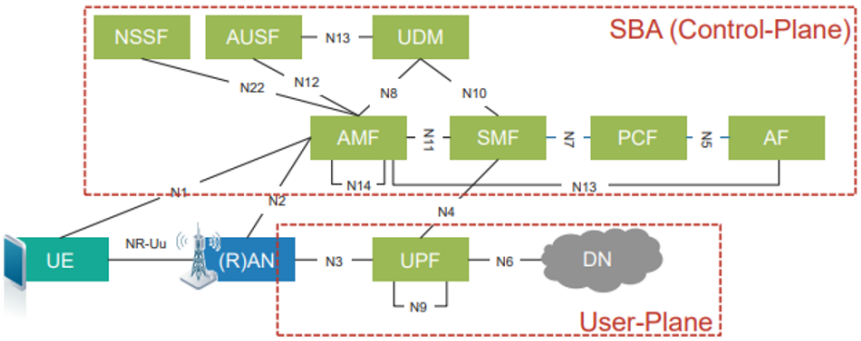

Figure 5.2: B5G network and its components.

platforms/devices. Examples of the software ones leverage the eXpress Data Path (XDP) [84] and the Data Plane Development Kit (DPDK) technologies. XDP is a framework based on the extended Berkeley Packet Filter (eBPF) technology that can run programs in a privileged context inside the system kernel; it enables the packet handling to bypass most of the operating system networking stack (e.g., context switches, interrupts, etc.). DPDK provides a poll mode driver that allows user-space applications to bypass the kernel and directly access dedicated network interfaces. Regarding the hardware accelerated UPFs, it is worth mentioning the P4 ones which have been studied in recent years [85]. Herein, I focus on the simplest implementation of UPF: a user-space application running inside a container. Moreover, said container is deployed only through the use of Docker rather than the popular Container Orchestration and management platform, Kubernetes, to avoid having to use additional plugins (i.e., Multus [86]).

5.3 Experimental validation

5.3.1 Experimental Setup

The full 5G network used to conduct the experiments on the UPF is provided in Figure 5.3. Both the Control and User Plane are deployed using the OAI opensource implementation [79]. In detail, the Control Plane NFs are deployed on a Kubernetes Cluster, while the UPF is deployed as a Docker container on a server with the Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz. As shown in Figure 5.3, the Access Network and Data Plane are simulated by Loadcore [80] agents. The Loadcore agents are deployed as VMs on a different server than the one under test.

The power consumption and performance of the UPF is measured by the well-known open-source power monitoring tool: Scaphandre [55]. By relying on Intel RAPL (Running Average Power Limit) [36], Scaphandre provides the power measurements of both the whole server and of every single process running on it. The power measurements are then saved on the standard de facto open-source timeseries database: Prometheus [58].

Regarding the performance of the UPF, the latency of the data plane traffic is considered as the performance metric. This metric is natively provided by Loadcore. The testing campaign involved five tests stressing the UPF while changing the enabled C_x states of each CPU of the server hosting the UPF. Each test was conducted with a different amount of C_x states enabled. Initially, only the active (i.e., C_0) state is enabled, then, progressively, the "sleeping" states are enabled. Therefore, in the last test all the C_x (i.e., { C_0, C_1,

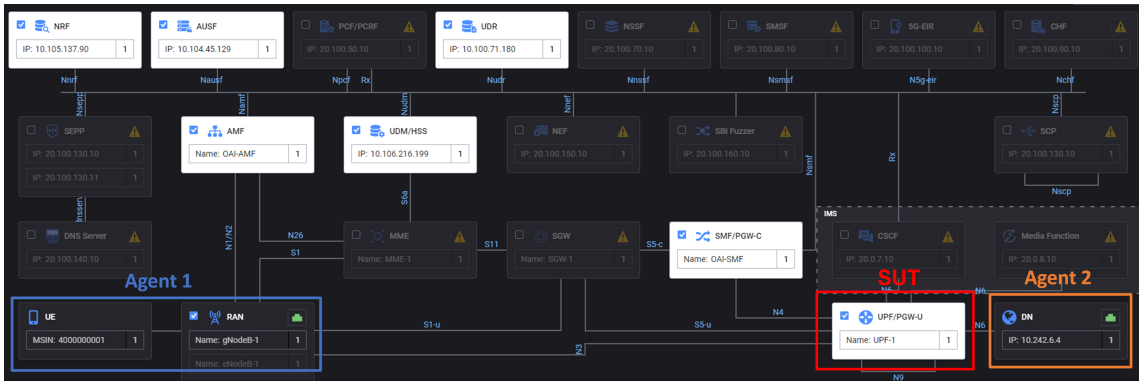

Figure 5.3: Details of the test setup as shown by the LoadCore dashboard. The NFs in the white boxes are provided by OAI, while the components encased in the blue and orange rectangle are simulated by two different LoadCore agents. The UPF, in the red rectangle, is the System under Test (SUT).

C_{1E}, C_3, C_6} can be assumed by every CPU of the hosting server. The enable/disable operation is carried out by a bash script that modifies the contents of the following folders: /sys/devices/system/cpu/cpux/cpuidle/statex. In each test, a single UE is attaching to the network and creating a single PDU session. The offered load rate in downlink and uplink ranges from 10 kbit/s up to 100Mbit/s with a 10-kbit/s step. The payload size is kept equal to 1250 bytes and only the packet rate is changed. For each rate, a test is conducted for a 5-minute duration followed by a 2-minute pause.

5.3.2 Experimental Results

Results concerning the power and performance of the UPF are presented in the following. In Figures 5.4 and 5.5 the UPF power consumption (in Watts) is shown. As can be noticed by looking at their x axes, the results obtained during the test campaigns are split into two graphs for the sake of representing them with the appropriate scale to have a better visual representation. Hence, in both graphs there are five power plots, one for each C_x states configuration of the physical server, and the secondary vertical axis represents the offered load (i.e., the traffic rate generated in downlink and uplink). It is worth highlighting that in Figure 5.4 (i.e., for lower offered loads), the power plots are very noisy and often reach 0 (especially in the 10 kbit/s case). This happens since, in order to obtain the correct offered load while keeping the payload size constant (equal to 1250 bytes), the packet rate is low; therefore, the UPF waits until its buffer is full enough before processing the packets.

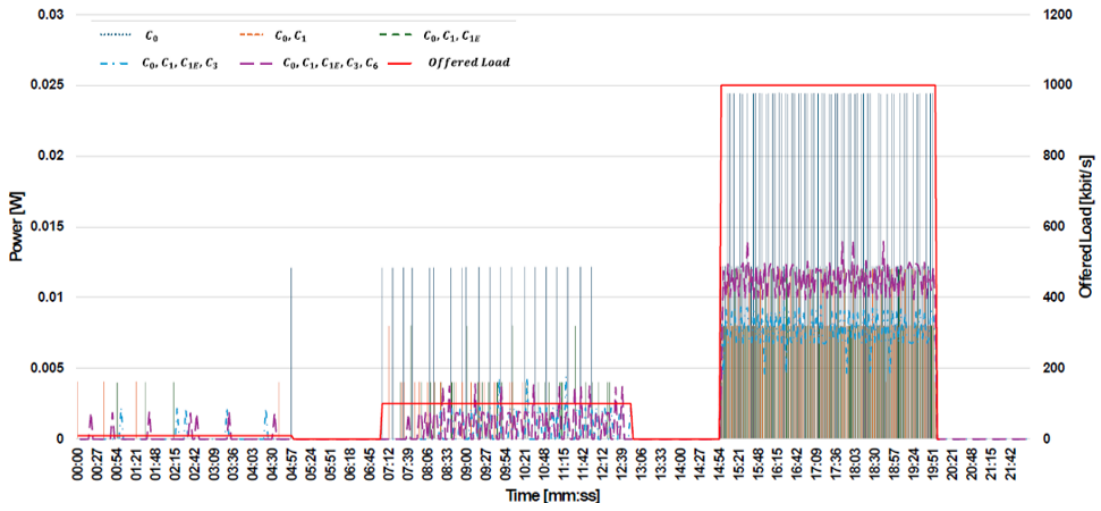

Figure 5.4: Power consumption of the UPF for offered loads ranging from 10 kbit/s up to 1 Mbit/s. Five power plots are shown: each one corresponds to a different combination of C_x states enabled.

Focusing now on the less noisy Figure 5.5, the power savings when deeper C_x states are enabled are examined. This is clearly shown by the test with the highest offered load (e.g., 100 Mbit/s), where the power is much lower in the purple plot (i.e., all C_x enabled) compared to the other ones. Moreover, the more C_x states are enabled, the lower the power, except for the combinations $\{C_0, C_1\}$ and $\{C_0, C_1, C_{1E}\}$; this is due to the fact that the C_{1E} is only a slightly enhanced version of C_1. However, focusing on the power plots with 10 Mbit/s as offered load, one can notice that the deepest state (i.e., C_6) causes only a tiny saving with respect to the active state and causes a higher power consumption than the other combinations. Therefore, for lower offered loads, the power needed to transition from C_0 and C_6 (and vice versa) hinders the expected savings.

In Table 5.1, the average UPF power consumption is reported. Again, it is worth noticing that, in the 100 Mbit/s scenario, enabling all the C_x states allows to save more than half of the power with respect to the always active (i.e., C_0) case. For other offered load, it is worth highlighting that enabling C_6 results in a higher power compared to the other C_x state combinations. So far, results show (for the most part) that, as the enabled C_x states deepen, the UPF power consumption decreases. However, this is not straightforward since I act on the hosting server rather than on the UPF container directly. As shown in Table 5.2, the power consumption of the hosting server decreases from left to right (e.g., from C_0 to $\{C_0, C_1, C_{1E}, C_3, C_6\}$) and from top to bottom (e.g., from lowest to highest

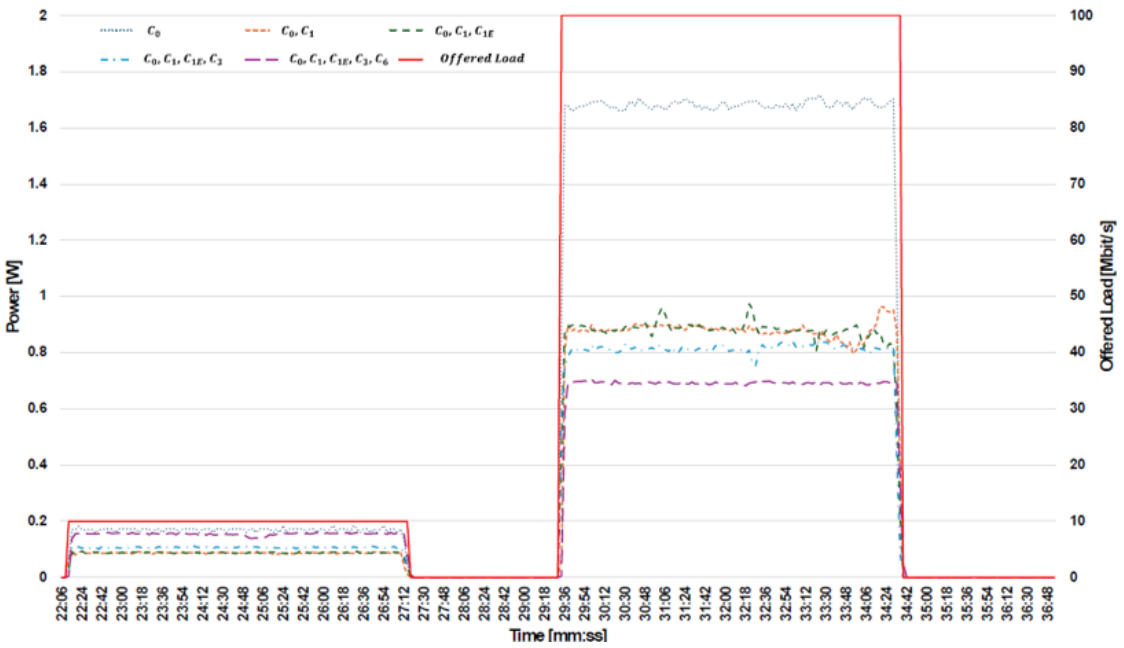

Figure 5.5: Power consumption of the UPF for offered loads equal to 10 and 100 Mbit/s. Five power plots are shown: each one corresponds to a different combination of C_x states enabled.

offered loads). However, since Scaphandre divides the total power of the server among the single processes based on their CPU usage, this explains how the change in host power is reflected onto the one ascribable to the UPF. Finally, the results regarding the UPF performance are shown in Table 5.3, which reports the average latency percentage of packets in the different C_x state combinations. It can noticed that no packets are lost in any scenario; thus, the UPF, in any C_x state-enabled scenario, would be able to route and forward mobile traffic without any drawbacks regarding the performance.

Table 5.1: Average power consumption (in Watts) of the UPF for the different offered loads and C_x state combinations.

Offered Load [kbit/s]	**C_0**	**C_0, C_1**	**C_0, C_1, C_{1E}**	**C_0, C_1, C_{1E}, C_3**	**$C_0, C_1, C_{1E}, C_3, C_6$**
10	0.0004	0.00020	0.00020	0.00013	0.00013
100	0.0025	0.00116	0.00128	0.00086	0.00107
1000	0.0184	0.00882	0.00916	0.00770	0.01117
10000	0.1692	0.08566	0.08663	0.10515	0.14981
100000	1.7025	0.85920	0.86945	0.80076	0.67467

Table 5.2: Average power consumption (in Watts) of the hosting server for the different offered loads and C_x state combinations. The first row provides the average power consumption for the pauses between each test.

Offered Load [kbit/s]	**C_0**	**C_0, C_1**	**C_0, C_1, C_{1E}**	**C_0, C_1, C_{1E}, C_3**	**C_0, C_1, C_{1E}, C_3, C_6**
0	203.91	67.18	69.03	36.26	31.75
10	199.35	68.56	67.15	37.40	31.34
100	203.63	67.19	67.17	37.15	31.48
1000	202.78	67.31	67.24	38.36	33.90
10000	206.70	67.92	68.22	53.58	48.88
100000	208.41	70.18	70.26	64.66	54.66

Table 5.3: Packet latency (in %) for the different C_x state combinations, averaged over the five offered loads tested.

Latency	**C_0**	**C_0, C_1**	**C_0, C_1, C_{1E}**	**C_0, C_1, C_{1E}, C_3**	**C_0, C_1, C_{1E}, C_3, C_6**
0us - 125us	99.75	99.57	99.42	99.45	99.41
125us - 250us	0.16	0.16	0.25	0.20	0.20
250us - 500us	0.05	0.25	0.27	0.29	0.29
500us - 1ms	0.03	0.01	0.05	0.05	0.07
1ms - 5ms	0.01	0.01	0.01	0.02	0.03
5ms - 10ms	0.00	0.00	0.00	0.00	0.00
10ms - 15ms	0.00	0.00	0.00	0.00	0.00
15ms - 20ms	0.00	0.00	0.00	0.00	0.00
20ms - infinite	0.00	0.00	0.00	0.00	0.00

5.4 Conclusions

In this chapter, I have examined the application of a well-known power saving technique for general-purpose servers on B5G networks. A classical power management technique has been used herein: the low power idle. It allows power saving caused by putting the CPU to "sleep" when not in use (when no packets are coming/being processed). The goal was to minimize the power consumption while not neglecting performance of the most power-hungry 5G NF: the UPF.

Tests were performed with different C_x (i.e., "sleeping") state combinations. The UPF was stressed by one UE continuously sending and receiving traffic with a varying rate. The testing environment was realized with a blend of opensource (for the 5G NFs and the monitoring stack) and commercial (for the Access Network and DN) solutions. Results show that, in most cases the deeper C_x states are enabled, the more power is saved. This is especially true in the 100 Mbit/s scenario in which the power is more than halved as a result. However, results show that special care should be devoted to the enabling/disabling of the "deepest" C_x state (C_6) since the transitions from it to the active state (C_0) (and vice versa) actually hinder the power savings.

Furthermore, the performance (as in latency) has been also evaluated. Results show no packet losses and no significant degradation. Thus, in all scenarios, mobile traffic can be supported without issue.

Finally, it is crucial to highlight that the obtained results hold for a specific implementation of UPF: the user-space application running on a container provided by OAI. As shown in [87], results are expected to change for other implementations of UPF. In further detail, the absolute values of the power consumptions are expected to change, while the saving trend based on the C_x is expected to hold. In any case, further investigation is needed to validate the results also with other implementations of the UPF.

Acknowledgements

This work has been partially supported by the Horizon Europe Research and Innovation Action 6Green (Grant Agreement no. 101096925), by the European Union under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, partnership on "Telecommunications of the Future" (PE00000001 - program "RESTART"),"SoBigData.it – Strengthening the Italian RI for Social Mining and Big Data Analytics" – Prot. IR0000013, and the European Union under NextGenerationEU. PRIN 2022 Prot. n. 2022MWBFEE.

CHAPTER 6

Machine-Learning Network Function Scaling via Black- and White-Box KPIs

6.1 Introduction

As mentioned in Section 2.4, horizontal scaling will play a crucial parte in 5G and Beyond. Herein, I focus on NF horizontal scaling devoted to energy efficiency by presenting a novel approach. Moreover, the method presented herein is an example of the analytics part of the framework introduced in Chapter 3.

Nowadays, scaling mechanisms heavily rely on AI and ML techniques for the autonomous, proactive, and cognitive management of the lifecycle of 5G applications and network functions. Most works in the state of the art on scaling decisions propose threshold-based solutions. Such thresholds can either be static (as in [44] and [46]) or dynamic (as in [45]). Static threshold solutions may lead to an oscillating pattern if the load frequently fluctuates around the thresholds. Moreover, in these solutions, few KPIs are analyzed to trigger the scaling decision. In particular, both [44] and [45] exploit only one metric, CPU utilization; in [46] CPU usage, RAM usage and Quality of Experience (QoE) are adopted. Among others, dynamic scaling for the automation of resource re-allocation is considered in [88], while the authors in [89] propose a system to dynamically scale stateful NFs by state disaggregation. The choice of the KPIs is especially critical because the link between the KPIs characterizing NFs/application components and CPUs is not straightforward, so that often one of the two categories is neglected. Nevertheless, their

The content of this chapter has been published in the following paper:

R. Bolla, R. Bruschi, F. Davoli, C. Lombardo, J. F. Pajo and B. Siccardi, "Machine-Learning-Based 5G Network Function Scaling via Black- and White-Box KPIs," 2023 21st Mediterranean Communication and Computer Networking Conference (MedComNet), Island of Ponza, Italy, 2023, pp. 143-150, doi: 10.1109/MedComNet58619.2023.10168859.

joint utilization would allow for a deeper understanding of the dynamics and efficiency of NFs/applications internal processes, and their dependency on the underlying hardware.

This Chapter presents a Deep Learning (DL) model, exploiting Artificial Neural Networks (ANNs), for the dynamic and automated scaling of NFs and applications according to the Zero-touch network and Service Management (ZSM) paradigm [90]. The model performs the automated addition or removal of NF/application instances across the geographically distributed edge network, according to real-time demands, and is based on a multiclass classification able to combine metrics on both the NF instance and the computational resources hosting it. The use of ML in this context provides two benefits: the ability to easily consider the contribution of a large number of KPIs and that of achieiving higher levels of accuracy, as it is not mandatory to divide the feature space in a linear fashion.

The model presented herein, based on recent specifications coming from the ETSI Experiential Networked Intelligence (ENI) standard [91], is not used merely for horizontal scaling, but it leverages on the slice-specific network and computing resource analytics provided by the NWDAF and the IDAF/MDAF for the automated allocation of NF/application instances in the zones with higher traffic density. In this work, the model is applied to the 5G SMF, which, being in charge of the establishment, modification and release of user sessions, can hugely benefit from the deployment of instances in the proximity of the UE.

Another novelty of this work lays in the generated dataset, which is a blend of both "black-box" KPIs (e.g., CPU and memory utilization) and "white-box" KPIs (e.g., NF metrics, such as the number of PDU sessions, etc.). Since the white-box KPIs are extracted from 5G standardized interfaces (via the aforementioned analytics) that do not depend on neither the application nor the underlying hardware, the model proposed herein is both application- and implementation-independent.

6.2 Methodology

6.2.1 Deep Learning Model

ANNs follow a bio-inspired approach by learning by example; as such, they have been gaining particular interest in modelling real-world systems, especially in the presence of noise and nonlinearity.

ANNs are composed of artificial neurons, also known as nodes or processing elements, arranged in layers: input, hidden, and output layers. Each artificial neuron receives input

signals, applies a mathematical transformation using weights and biases, and produces an output signal passed to neurons in the next layer. The weights determine the strength of the connection between neurons, while the bias adjusts the output of the activation function applied to the weighted sum of inputs. Activation functions such as sigmoid, Rectified Linear Unit (ReLU), and hyperbolic tangent (tanh) introduce nonlinearity, enabling ANNs to model complex relationships in data. The output of each neuron is presented as a connection strength to all neurons of the next layer. The output of the neuron(s) in the output layer is a number ranging from –1 to +1 when using specific activation functions [92].

Common ANN architectures include feedforward networks, where information flows in one direction from input to output without feedback; Multilayer Perceptrons (MLPs), which are feedforward networks with one or more hidden layers; Recurrent Neural Networks (RNNs), which incorporate feedback connections for processing sequential data; Convolutional Neural Networks (CNNs), which use convolutional and pooling layers for grid-like data such as images; and Deep Neural Networks (DNNs), which have multiple hidden layers to enhance predictive capabilities. In CNNs, convolutional layers employ filters to scan input data, pooling layers reduce spatial dimensions, and fully connected layers map extracted features to outputs. RNNs are designed for sequential or time-series data, with connections that allow information to persist across time steps. DNNs stack several processing layers, such as dense, convolutional, attention, or recurrent layers, to automatically extract high-level features from raw data [93].

There are two main approaches for ANN-based predictive modelling working in a supervised fashion:

1. Classification which predicts a category.

2. Regression which predicts a quantity.

The former approach is the one adopted in this work. While the latter could allow introducing traffic forecasting to anticipate the traffic needs with the goal of realizing a proactive scaling mechanism allowing for the reduction of the time between the need for scaling and the actual action. Both approaches aim at learning the function that best maps the input variables to the output variables (either as a category or as a quantity) through nonlinear approximations.

In this section, I outline the classification problem in the context of horizontal service scaling that will be used as basis for the ZSM solution's DL model. In more detail, a model

is built based on a training dataset (i.e., historical data), which is generally organized into a table of N rows (i.e., samples) and $M + T$ columns (i.e., M features/input variables and T target/output variables); note that in the current example $T = 1$. The number of features defines the number of neurons in the input layer. For a classification problem, the number of neurons in the output layer is defined by the number of classes/labels C (for instance, there are three classes in the sample problem – ScaleIN, Maintain and ScaleOUT – hence, $C = 3$); for a regression problem, the number of neurons is determined by the number of targets (T). Between these two layers, there can be one or more hidden layer(s) with a variable number of neurons.

In DL applications, the deeper the ANN becomes, the more complex problems it will be able to solve, at the cost of increased computational complexity. Regarding the anatomy of a neuron, the inputs are used to obtain a biased, weighted sum that is then fed to the activation function to produce the output. Some of the most commonly used activation functions include the ReLU, sigmoid, softmax and hyperbolic tangent. The optimal ANN structure (i.e., the number of hidden layers, neurons per layer and their corresponding activation functions) highly depends on the application and the dataset. This decision is usually based on experience, search strategies [94], Genetic Algorithms [95], k-fold cross validation [96], etc.

The model used in this paper is a feed-forward back-propagation ANN, trained by iteratively updating the weights and biases to minimize the error between the predicted and expected target values in the dataset. Starting with a random initialization, predictions are calculated by feeding the data forward through the network. Then, computed according to a specific loss function (e.g., cross-entropy, mean absolute error, mean squared error, among others), the error is propagated backwards. The weights and biases are updated according to the chosen optimizer and learning rate. This forward-backward propagation process is repeated until the loss function converges or a specified maximum number of iterations is reached. The schema of an example of a feed-forward back-propagation ANN is shown in Figure 6.1.

Open-source tools (such as scikit-learn [97], TensorFlow [98], Keras [99] and PyTorch [100]) are already widely available for building DL models in Python. In this work, recent versions of TensorFlow (which already has a tight integration with Keras) are considered to jointly benefit from the high-level APIs of Keras and the low-level control of TensorFlow.

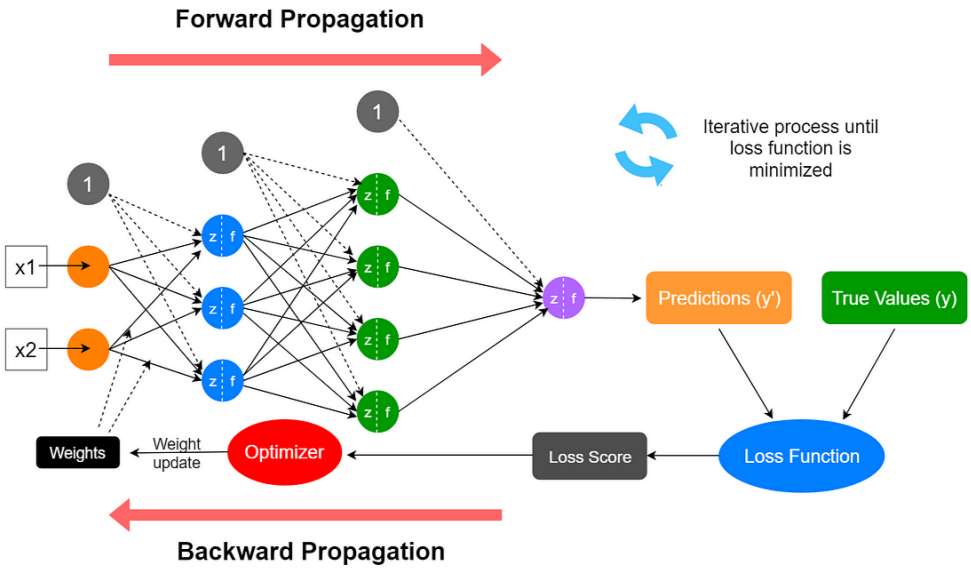

Figure 6.1: Schema of a feed-forward back-propagation ANN [9].

6.2.2 Service Scaling with Deep Learning

The proposed model exploits deep ANNs for the zero-touch orchestration of a NF/application instance and is meant to be introduced in ETSI ENI-compliant network platforms and fed by the metrics and analytics from the NWDAF and IDAF/MDAF (see Section 4.2.1). The outcome is the automated addition or removal of the instances across the geographically distributed edge network according to the real-time demands. In particular, the scaling problem is modelled as a Multiclass Classification problem that seeks to categorize various high-/low-level metric combinations into "superstates" relating to scaling decisions.

Let $\mathbf{X}$ be the set of (potentially) relevant high-/low-level metrics, and Y be the scaling decision. The goal is to find the function f that best maps $\mathbf{X}$ to Y, based on a training dataset that consists of historical $(\mathbf{X}, Y)$ combinations. In a multiclass classification problem, Y refers to the label that corresponds to the "superstates" that categorize the metric combinations into classes of scaling decisions, such as whether to scale in, maintain or scale out for a given sample of $\mathbf{X}$.

Given a training dataset, a ML classification algorithm seeks to divide the input space into decision regions corresponding to the C discrete classes, and builds a model by creating decision boundaries between these regions based on the statistical patterns in

the dataset [101] [102]. It is important to note that the classes of a given dataset may not always be exactly separable; hence, the goal of the algorithm is to find the model f that optimizes the cost function (such as maximizing accuracy or minimizing loss, among others).

Suppose that the random variables $\mathbf{X}$ and Y evolve over time depending on the service demands, and each combined realization or sample $(\mathbf{x}^{(t)}, y^{(t)})$in the training dataset is indexed in temporal succession, $t = 0,\ 1, \ldots$. In a probabilistic view of multiclass classification, the model f and its parameters are found by maximizing the likelihood function $\mathcal{L}(\{P\left(y^{(t)}|\mathbf{x}^{(t)}\right),\ t = 0,\ 1,\ \ldots\})$ or, equivalently, minimizing the loss function, $E = -\log(\mathcal{L})$ during the training phase, such that f best captures the statistical patterns between the samples [101] [94].

Once a model has been built, it can then be used to automatically classify unlabelled samples of $\mathbf{X}$, for instance, with an independent test dataset or in an actual production environment. During these phases, the model computes the posterior probabilities for each class, $P\left(y = c|\mathbf{x}\right),\ c = 1, \ldots,\ C$, given the unlabeled input data, and the class corresponding to the highest $P\left(y = c^*|\mathbf{x}\right)$ indicates the predicted label for each sample.

An ANN-based classifier is considered in this work to capture the statistical patterns between $\mathbf{X}$ and Y, in a supervised learning approach. This choice is mainly motivated by the ANN's ability in modelling real-world systems, especially in the presence of noise and nonlinearity. As described in [102], an ANN-based model is a "nested" mathematical function f_{ANN} such that for an L-layer network:

$$Y = f_{ANN}(\mathbf{X}) = f_L(\cdots f_2(f_1(\mathbf{X}))) \tag{6.1}$$

where cases with $L > 2$ (or with more than one hidden layer between the input and output layers) are considered as deep ANNs. Generally, for layer $l = \{1, \ldots,\ L\}$,

$$y_l = f_l\left(\mathbf{z}\right) = g_l(\mathbf{W}_l\mathbf{z} + \mathbf{b}_l) \tag{6.2}$$

where $\mathbf{z}$ is the output of the preceding layer, g_l is the activation function, $\mathbf{W}_l$ is the weight matrix and $\mathbf{b}_l$ is the bias vector. The rows in the matrix $\mathbf{W}_l$ are vectors $\{\mathbf{w}_{l,m_l}, m_l = 1, \ldots,\ M_l\}$, each with the same dimensionality as $\mathbf{z}$, where M_l is the number of neurons in layer l.

It can be noted how the output layer generates a probability distribution $\{P\left(y = c|\mathbf{x}\right),\ c = 1, \ldots,\ C\}$, based on which the predicted label $\hat{y}$ can be obtained as the one corresponding to the highest $P\left(y = c^*|\mathbf{x}\right)$ value.

Most of the model parameters shown above are not solely learned from the training samples, but also from the outputs of the preceding layers, especially in DL applications [102]. In this view, a feed-forward backpropagation ANN starts with a random initialization of the model parameters and calculates the predictions (e.g., with a Maximum Likelihood criterion) by feeding the data forward through the network; it then computes the error according to a specific loss function (e.g., cross-entropy or negative log likelihood) and propagates it backwards to update the weights and biases. As pointed out in [101], the learning process involves solving a nonlinear optimization problem and evaluating the derivatives of the loss function with respect to the model parameters; backpropagation is a technique for efficiently evaluating the gradient of the loss function for training feed-forward networks.

The most popular choices of activation functions in recent works are the ReLU and the Softmax functions. ReLU is typically used in the hidden layers, while the activation function used in the output layer highly depends on the application – for instance, for a multiclass classification problem, Softmax is typically used [102] [94]. As previously anticipated, training an ANN entails solving an optimization problem – that is, minimizing the loss function. This can be performed through a variety of optimization algorithms, or the so-called "optimizers". In this work, we use the Adaptive Moment Estimation (Adam) [103].

6.2.3 Machine Learning Workflow Applied to the 5G Environment

This section describes how the service scaling model introduced in Section 6.2.2 is applied to the 5G context.

ML workflows usually start with Data Collection from one or more data sources, carefully selected according to their potential relevance to the problem at hand. Since the acquired (raw) data may have different formats, especially if they come from several sources, the raw data need to undergo Pre-processing, in which they are cleaned and formatted into a usable dataset, suitably structured for the problem; feature analysis, label/target definition and splitting of the dataset (into training and test sets) are also done in this phase.

Once the structured dataset is ready, then comes the Modelling phase – where the training set is fed to the algorithm to learn the model parameters; in some cases, a validation set can be used to further refine (e.g., fine-tuning of parameters) the model. In

the Deployment phase, the resulting model is tested with an independent test set, and/or in a production environment.

6.2.3.1 Data Collection

Let me consider the presence of a Prometheus database [58] which maintains the time series that the NWDAF and the IDAF/MDAF can obtain from the SMF on the connection and mobility management tasks and on the NF usage statistics, respectively. Both the IDAF and MDAF provide information related to the infrastructure and management plane; however, in this case the black-box KPIs originate from the IDAF (i.e., cpu usage), rather than the MDAF. A possible extension of this work can concern the use of other KPIs (e.g., the power consumption generated by the MDAF).

The testbench used to generate the dataset and validate the model is shown in Figure 6.2. In particular, for mimicking the presence of a 5G network, from the access (including the data generated by the users) to the core, I use UERANSIM [104] and Free5GC [105], respectively. The former is a Linux-based emulator of a complete 5G access network, including UEs and gNodeBs, while the latter provides a complete 5G core with the individual NFs provided in a containerized fashion, including our targeted SMF. Both tools are open source; UERANSIM is deployed through multiple Virtual Machines (VMs)

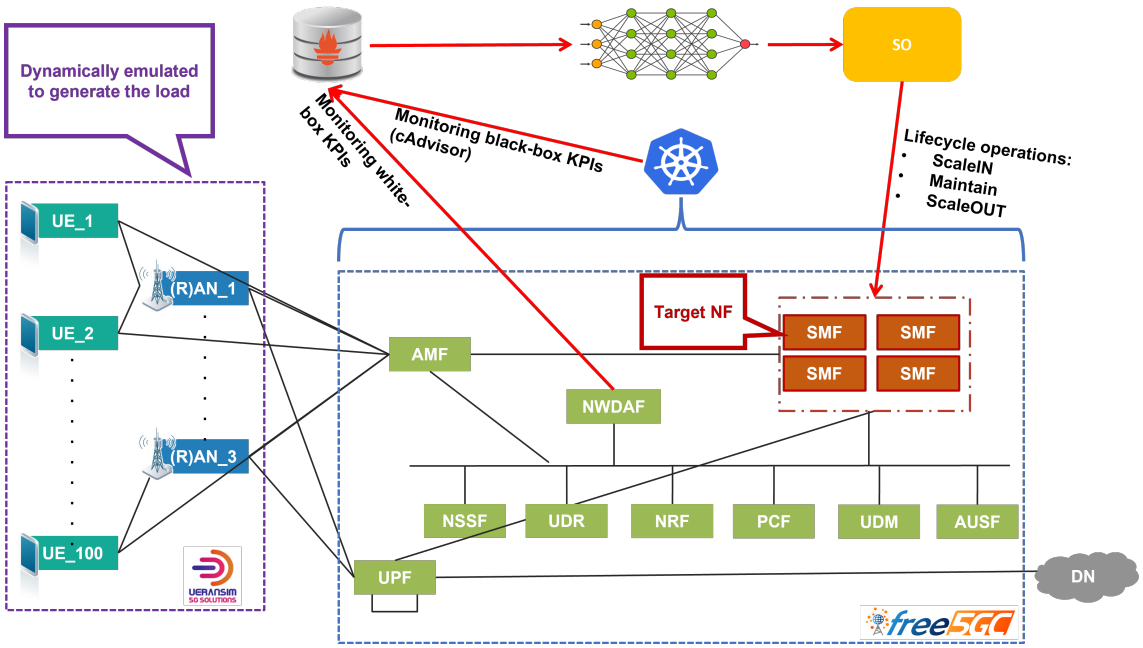

Figure 6.2: Testbench used to generate the dataset.

and Free5GC relies on Kubernetes [47]. In each test, the number of UEs is randomly selected between 0 and 100 and gNodeBs vary between 1 and 3, resulting in varying system workloads. Furthermore, the 5G core network is container-based and runs on top of a Kubernetes cluster. Each SMF instance has resources limited to one vCPU and 512 MB of RAM. This configuration has been chosen to reach the maximum capacity of an SMF in the tests, and hence, the system will need to scale out or add another SMF instance.

The tests have been performed in an offline environment, such that each test has a corresponding Prometheus export which contains both the white-box and the black-box KPIs. In order to retrieve the former and passing them to Prometheus, since the current support of the REST APIs for events subscription in Free5GC NFs is incomplete, I extended the NWDAF to access the SMF logs (via the Loki framework). The latter are exported from Kubernetes to the Prometheus database through cAdvisor [52] which provides data regarding resource usage and performance characteristics on a container basis. The KPIs are exported to Prometheus every 1 second. The Prometheus data generated for the aforementioned tests are parsed to obtain the necessary features and data samples, then aggregated according to the desired structure. As previously mentioned, in order to generate a usable dataset, the data need to be carefully aggregated with consistent timeframes, and then, suitably structured for the problem at hand.

6.2.3.2 Pre-processing

By looking at the (created, responded, closed) timestamps exported from Prometheus, 8 white-box KPIs can be derived per second: number of initiated and released Protocol Data Unit (PDU) sessions, number of allocated QoS Flow Identifiers (QFIs), user plane path changes, N10 and N11 terminations, tracking areas and Non-Access Stratum (NAS) interactions. From the data exported by cAdvisor, 6 black-box KPIs from the CPU utilization (user, system, total) and from the memory utilization (rss, working_set, total) are considered. The black-box metrics are further processed to obtain the average, min and max values per second across all active SMF instances, which results in a total of 32 features.

Data cleaning usually involves handling missing and/or invariant data. For instance, in the case of missing data, the distribution of the missing values can be initially checked; if there are only few missing values, one can either drop the samples or use the average value of the samples; otherwise, drop the columns with many missing values. After cleaning, the number of input features to be used in the actual training/testing is reduced from 32 to

24 features. Given the structured (and cleaned) dataset, it can now be split into training and test sets; the former can also be further split for the actual training and the intermediate validation. In this work, the training set comprises 60% of the samples (20% of which was reserved for the validation in the model fitting), and the remaining 40% was used in the final testing.

Let me recall that the three classes – ScaleIN, Maintain and ScaleOUT – act as "superstates" classifying the metric combinations towards the decision of whether to scale in, maintain or scale out the SMF instances. The input features in the dataset characterize the historical metric combinations, while the corresponding labels characterize the scaling decisions, where the latter is usually based on technical expertise.

Since the initial dataset is not labelled, we suppose that the historical scaling decisions are governed by some conditions based on the CPU utilization and the number of active instances, and generate the labels accordingly. However, it is important to note that in real-world ML/AI applications such conditions are supposed to be unknown to the developer, and learned by the model, on the basis of the given features and labels of the dataset. The sole purpose of the following conditions is to generate a working dataset.

In more detail, 1-minute moving averages of the CPU utilization (= $\%total$) of each SMF instance are obtained and the mean across all active instances is considered:

$$\overline{MovAvg\_CPU\_util}^{(t)} = \frac{\sum_{\tau=t-T-1}^{t} \sum_{\upsilon=1}^{V^{(\tau)}} \left(\%total_{\upsilon}^{(\tau)}\right)}{T \times V^{(t)}} \tag{6.3}$$

where $T = 60$ (seconds) and $V^{(t)}$ is the value of $\#SMFs\_active$ at time instant t. Based on the runtime values of $\overline{MovAvg\_CPU\_util}^{(t)}$, the supposed conditions are shown in Figure 6.3.

To better illustrate the service scaling scenario, Figure 6.4 shows how $\overline{MovAvg\_CPU\_util}$ and $\#SMFs\_active$ vary with time, as well as the desired scaling decision at each instant. It can also be observed that the notifications for the need of scaling in/out SMF instances continue until the service orchestrator actuates the desired decision.

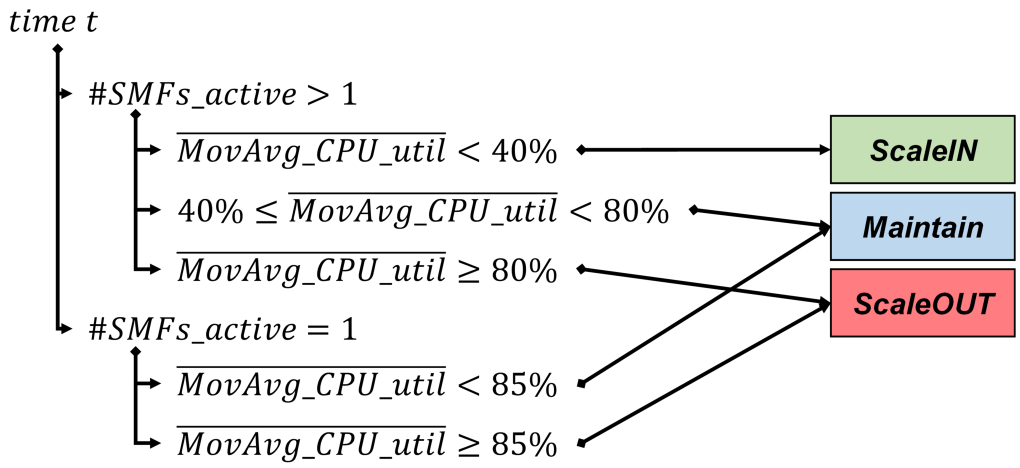

Figure 6.3: Example of service scaling conditions.

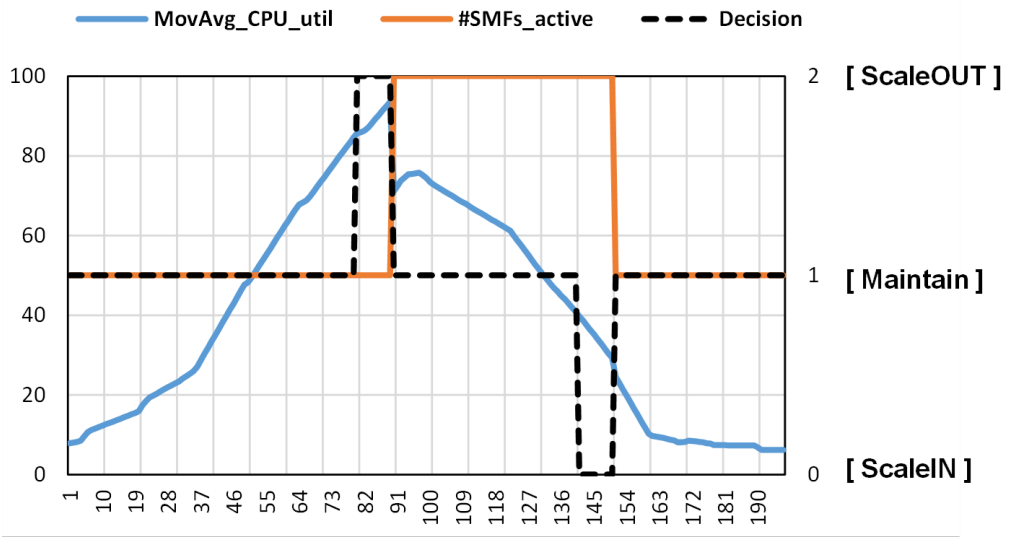

Figure 6.4: Example of service scaling decisions based on CPU utilization and number of active instances.

6.3 Experimental validation

6.3.1 Model Training and Validation

In this work, the widely used TensorFlow-Keras bundle is adopted for building the ANN-based model. With Keras, ANNs can be constructed in a modular and fully configurable way using its built-in classes for various components (such as the layers, loss functions,

activation functions and optimizers, among others). It also features automatic backpropagation that simplifies the training process for the users.

A Sequential model [106] is considered for the multi-class classification problem at hand, and it is built by basically stacking ANN layers, each with exactly one input and output tensor (i.e., multi-dimensional array). Dense layers [107] are used in the model – particularly, 2 fully-connected hidden layers with 24 neurons/layer (i.e., set to equal the number of input features, dataset.shape[1]–1) and ReLU activation functions, then 1 fully-connected output layer with 3 neurons (i.e., set to equal the number of classes) and SoftMax activation functions. Adam is chosen as optimizer, with a learning rate η of 0.01, and default exponential decay rates (i.e., $\beta_1 = 0.9$ and $\beta_2 = 0.999$) [103]. Lastly, sparse categorical crossentropy and sparse categorical accuracy are used as loss function and metric, respectively, since the labels are not one-hot encoded and kept their integer values {0, 1, 2} (i.e., as on the right in Figure 6.4).

Now, recalling the anatomy of the neuron, with a dense layer (fully connected), the number of inputs received by each neuron is equal to the number of neurons in the preceding layer, and its output is therefore also received by all neurons in the succeeding layer. Moreover, each neuron has one bias parameter. With this in mind, each hidden layer in the model constructed above has a total of 600 (= 24 * 24 + 24) trainable parameters, while the output layer has 75 (= 3 * 24 + 3). All in all, the model has 1275 trainable

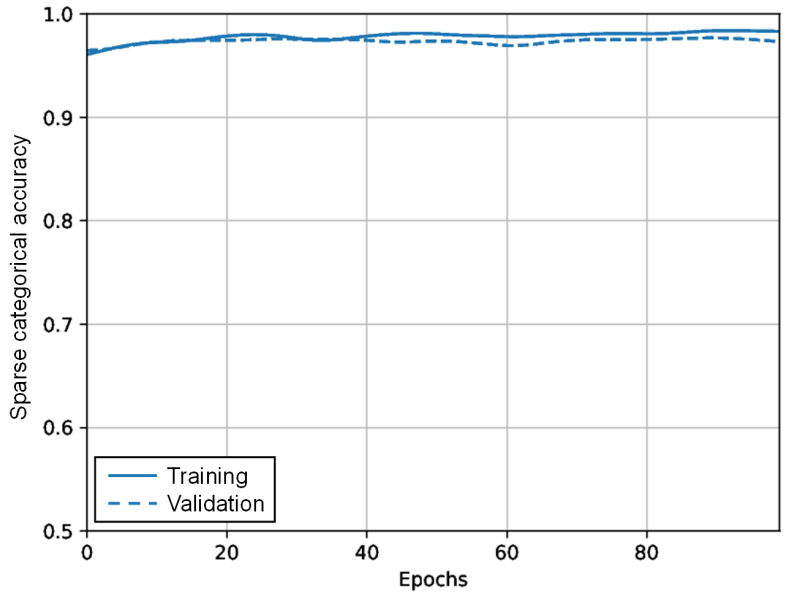

Figure 6.5: Training and validation results.

parameters. The training set used as input to the model has 4129 (60% of the total) samples – 80% of which are used for actual training, while the remaining 20% is used for validation. Figure 6.5 shows the training and validation accuracies obtained in 100 epochs. It can be observed that the model has stable accuracies at around 97~98%. It is also interesting to note that with 3 hidden layers, such accuracies start to decrease with increasing epochs, owing to overfitting. Finally, it is worth mentioning that the model was trained on a VM with 4 vCPUs and 8 GB of RAM; with this configuration the training phase lasted 30.48 seconds.

6.3.2 Model Testing and Deployment

This phase comprised the testing of two models: a threshold-based model and the ANN model thoroughly explained in this Chapter. The former is a static threshold model that is based on the one presented in [46].

In particular, thresholds are applied to both the black-box and the white-box KPIs; eventually the result is decided by an OR operator between the two sets. In both cases, 2753 samples (40% of the total) were used in this phase; these samples were never-before-seen samples, in order to reflect an actual production environment.

One way of showing the performance of a classification model is through a confusion matrix [102], which summarizes the success of predicting the classes of the test samples, with one axis indicating the true label and the other the predicted label. Figures 6.6 and 6.7 show the confusion matrices obtained with the two aforementioned models. In both cases one can observe the confusion matrices in terms of both the number of samples (Figures 6.6(a) and 6.7(a)) and the percentages (Figures 6.6(b) and 6.7(b)). By comparing the results shown in Figures 6.6(b) and 6.7(b), it can be observed that the ANN-based classifier has a higher accuracy than the threshold-based one; in particular, the former was able to correctly classify 99.62%, 95.29%, 97.06% of the samples belonging to the classes $\{0, 1, 2\}$, respectively.

It can also be noted in Figure 6.6(a) and 6.7(a) that the sample set belonging to class 2 is substantially small (~10%) compared to the other two classes; hence, the dataset is class imbalanced. This can provide indications on which performance metrics to look at.

Starting from the confusion matrix, different performance metrics can be derived – namely, the Overall Accuracy, Precision, Recall and F1-score. The Overall Accuracy is given by the ratio of the number of correctly classified samples to the total number of samples.

$$Overall\ Accuracy = \frac{\sum_c TP(y=c)}{\#\ of\ test\ samples} \tag{6.4}$$

While this is a great metric for symmetric datasets, it is not the proper measure for class imbalanced datasets. In such cases, the other three metrics are usually adopted. Precision is given by the ratio between the number of true positive (TP) predictions and the total number of positive predictions (i.e., TP) plus the false positive (FP) predictions.

$$Precision(y=c) = \frac{TP(y=c)}{TP\,(y=c) + FP\,(y=c)} \tag{6.5}$$

This is a great metric for cases when the cost of FPs is high, such as in service scaling where FPs resulting in the over-provisioning of resources generate higher operational costs [94]. Recall is given by the ratio between TPs and the total number of positive test samples (i.e., TP plus the false negative (FN) predictions) [101].

$$Recall(y=c) = \frac{TP(y=c)}{TP\,(y=c) + FN\,(y=c)} \tag{6.6}$$

In contrast to Precision, this is a good metric for cases when the cost of FNs is high, such as in service scaling where FNs resulting in the under-provisioning of resources generate (potentially severe) degradations to the quality of service (QoS). Finally, the F1-score is the weighted average of the last two performance metrics, which is given by:

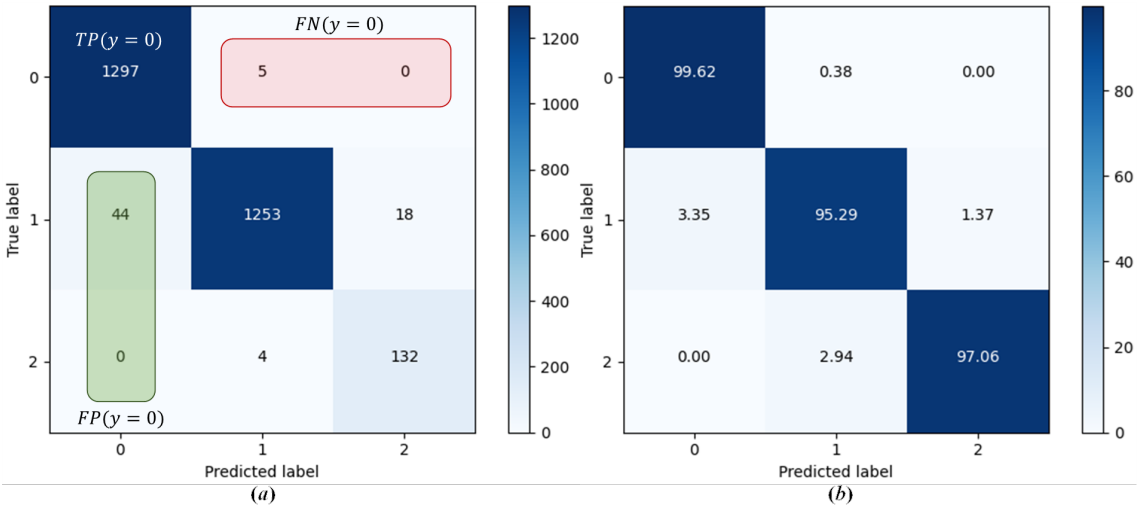

Figure 6.6: Confusion matrix of the ANN-based model in terms of: (a) the number of samples, and (b) percentage.

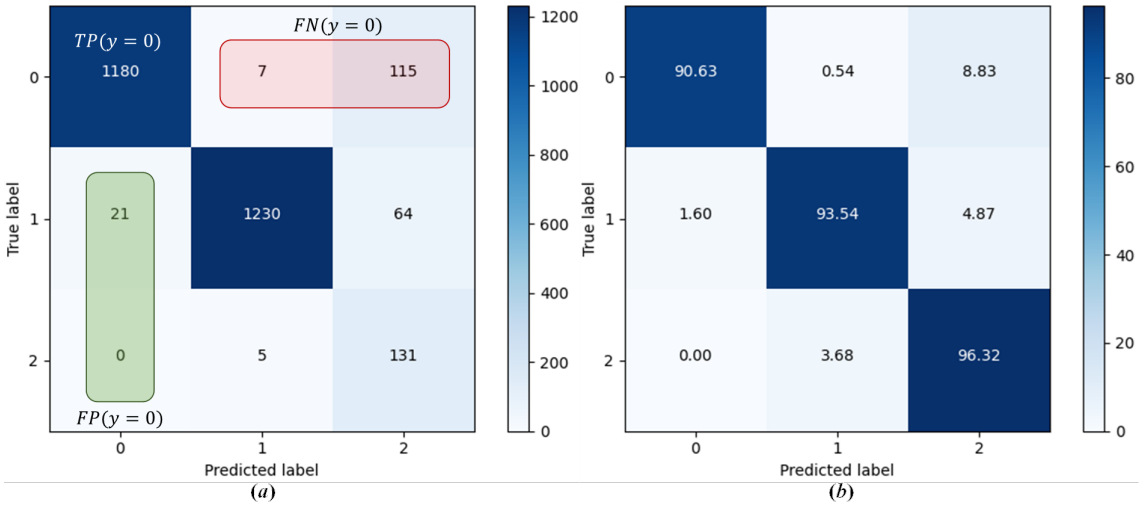

Figure 6.7: Confusion matrix of the threshold-based model in terms of: (a) the number of samples, and (b) percentage.

$$F1\text{-}score(y=c) = \frac{2 * Precision(y=c) * Recall(y=c)}{Precision(y=c) + Recall(y=c)} \tag{6.7}$$

Figure 6.6(a) and 6.7(a) indicate the samples giving the number of TPs, FPs and FNs for class 0.

Table 6.1 shows the classification report of the ANN-based model for the independent test set, indicating the values for the Overall Accuracy and the per-class Precision, Recall and F1-score. It can be observed that the values of all the performance metrics are high; class 2 obtained the lowest performance (though still at around 88%~97%) due to the small number of samples available in both training and testing. Finally, it is worth mentioning that the overall accuracy of the threshold-based model is equal to 0.923; therefore, the ANN model's one is ~5% higher.

Table 6.1: Classification report of the ANN model.

Class	***Precision***	***Recall***	***F1-score***	***Samples***
0 (ScaleIN)	0.967	0.996	0.981	1302
1 (Maintain)	0.992	0.953	0.972	1315
2 (ScaleOUT)	0.880	0.971	0.923	136
Overall accuracy: 0.974				2753

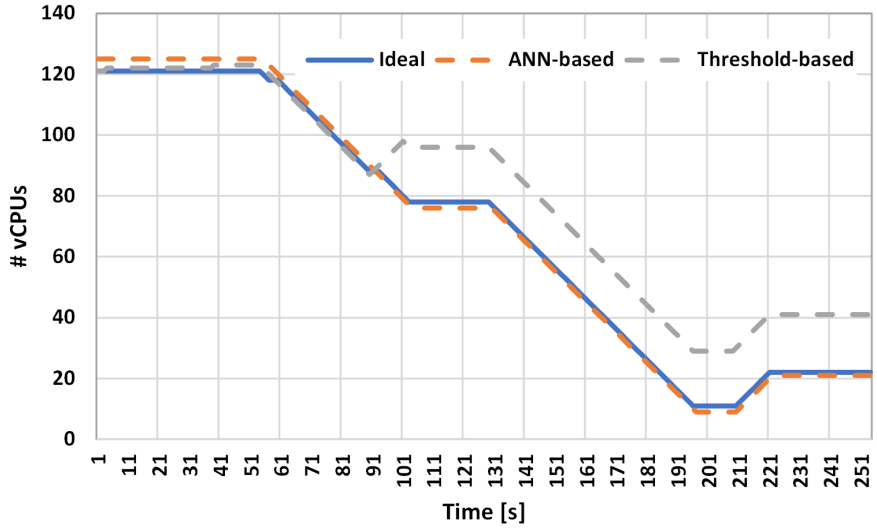

Figure 6.8: Number of allocated vCPUs in time for 254 test samples taken randomly from the test set for three cases.

Finally, recalling again Figures 6.6(a) and 6.7(a), the threshold-based classifier has a significant lower classification accuracy (by ~9%) in the 0 class which corresponds to the scaleIN action. Therefore, by using the ANN-based classifier, one can avoid over-provisioning, which consists of the allocation of unneeded resources. This can be further observed in Figure 6.8 which shows the plots of the allocated vCPUs in time for 254 test samples taken randomly from a uniform distribution among the test set. In Figure 6.8, three plots are shown; they represent respectively the allocated vCPUs for the ideal, the threshold-based and the ANN-based case. Regarding the DL model, it can be observed that initially the allocated vCPUs are higher than in the other two cases; this is due to the resources devoted to the ANN classifier, which are added to those needed by the SMF instances. On the contrary, in the threshold-based model the added resources to classify the samples were not considered, since it requires a negligible computational effort. The actuation time is not considered since it is the same in both models (ANN- and threshold-based). Furthermore, in the ANN-based plot the delay due to the sample classification is considered; however, it cannot be noticed from the figure, since the delay is equal to 0.132 milliseconds, which is a lot smaller compared to the interval between samples (1 second). The ideal plot represents the number of allocated vCPUs in the case in which the classification resources are negligible and in which the samples are always classified correctly. Comparing the three plots, it can be observed that the ANN-based model allows to save a significant amount of resources (20 vCPUs maximum in this example)

compared to the threshold-based one and that it closely follows the ideal plot except for some classification errors.

6.4 Conclusions

In this Chapter I proposed a DL model, which exploits deep ANNs, to drive the dynamic and automated scaling of 5G applications and network services. Thanks to the usage of both standard black-box (execution environment) and white-box (standard 5G APIs) KPIs, the proposed model is independent of specific 5G NF implementations. Results have been obtained by applying the proposed model on an SMF and have shown that the training and validation phases reach an accuracy of 97~98%, with the deployed model going above 95% for most numbers of available samples, proving higher overall accuracy by ~5%, and computational resource savings with respect to a threshold-based one.

This work can be further extended by turning it into a proactive scaling mechanism. This can be achieved by the introduction of traffic forecasting (e.g., regression) techniques, through which it would be able to anticipate the traffic needs. The benefit of a proactive scaling mechanism lies in the reduction of the time between the need for scaling and the actual action. Recalling Section 2.2, another possible extension of this work consists of adding a new black-box KPI: the power consumption of the NF. This addition would contribute to fostering greener B5G technologies.

Acknowledgements
This work has been supported by the Horizon 2020 5G-PPP Innovation Action 5G-INDUCE (Grant Agreement no. 101016941).

CHAPTER 7

General conclusions

This thesis addressed the issue of power efficiency in B5G networks. The advantages of the newest mobile network technology (5G) were exploited, while the drawbacks were tackled.

The mandatory starting point was the power measurement of 5G network components. Thus, I started by presenting a flexible Observability, Monitoring and Analytics framework. On the one hand, it handles the observability part (i.e., data collection and simple operations) and, on the other hand, it handles the analytics part which consists in complex algorithms that perform any kind of analysis (e.g., statistical dependance, classification, forecasting, etc.). The other crucial starting point was studying and comparing the already avaible tools to monitor the power consumption of virtualized applications (e.g., VMs and containers); this led to finding Scaphandre as the most suitable one.

The aforementioned framework was integrated and tested inside a 5G network. Along this, I presented a prototype of MDAF which deals with data coming from the management plane. The proposed MDAF provides a 5G NF power measurement that takes into account both the physical resource (i.e., CPU, RAM, disk, network I/O) consumption and the power necessary to keep the OS, the potential hypervisor and/or the container management orchestrator up and running. The MDAF was fully tested by using both opensource software and commercial ones; tests were performed on the User-Plane and on the Control-Plane. Results showed that the MDAF is able to detect a higher power consumption than Scaphandre which could become significant in case of NFs deployed in multiple instances. Moreover, results show that most of the difference in power comes from operations related to the container management orchestrator and the scheduler of the OS.

Another contribution of this thesis concerns the application of the well-known power saving technique commonly used in general-purpose server: LPI. It entails putting the unused CPUs to "sleep" when no tasks need to be performed. The degree of "sleep", and therefore of power saving and time to wake the CPU up, depends on the x of the C_x state

assigned to the CPU. This technique was tested on a 5G network and, in particular, on the UPF. Results show that, in most cases (all, except C_6) the deeper C_x states are enabled, the more power is saved. Moreover, the power saving didn't come at a cost for performance: results showed no packet losses.

Finally, an ANN-based algorithm was presented as a good fit for the Analytics block in the overall framework presented at the beginning of the thesis. In detail, the ANN model feeds on both standard black-box (e.g., CPU usage) and white-box (e.g., number of active UEs) KPIs and drives an NF scaling decision on the SMF instances. Thus, the stated ML problem regards a multiclass classification problem where the classes correspond to scaling decisions: scaleIN, maintain and scaleOUT. The proposed model was tested in a 5G environment and compared to a standard threshold-based one. Results showed a 97~98% accuracy in the training and validation phases and an overall accuracy of more than 97% on never seen before samples. Being the ANN-based model accuracy higher (9% at the maximum) than the threshold-based one, computational resource usage is decreased.

Publication record

1. International Journal Papers

[1] R. Bolla, R. Bruschi, C. Lombardo and B. Siccardi, "6G Enablers for Zero-Carbon Network Slices and Vertical Edge Services," in IEEE Networking Letters, doi: 10.1109/LNET.2023.3262861.

[2] M. Akbari, R. Bolla, R. Bruschi, C. Lombardo, N. S. Martinelli, B. Siccardi, A Management Data Analytics Function for ethical 6G networks, Computer Networks, Volume 270, 2025, 111487, ISSN 1389-1286, https://doi.org/10.1016/j.comnet.2025.111487.

2. International Conference Papers

[3] R. Bolla, R. Bruschi, C. Lombardo S. Mangialardi, A. Mohammadpour, R. Rabbani and B. Siccardi, "A Multi-Tenant System for 5/6G Testbed as-a-Service," 2023 15th International Conference on COMmunication Systems & NETworkS (COMSNETS), Bangalore, India, 2023, pp. 768-773, doi: 10.1109/COMSNETS56262.2023.10041360.

[4] R. Bolla, R. Bruschi, F. Davoli, C. Lombardo, J. F. Pajo and B. Siccardi, "Machine-Learning-Based 5G Network Function Scaling via Black- and White-Box KPIs," 2023 21st Mediterranean Communication and Computer Networking Conference (MedComNet), Island of Ponza, Italy, 2023, pp. 143-150, doi: 10.1109/MedComNet58619.2023.10168859.

[5] R. Bolla, C. Lombardo, P. Bono, N. S. Martinelli, R. Bruschi and B. Siccardi, "An Open-Source Prototype of Network Data Analytics Function for Next-Generation 5/6G Environments," 2023 IEEE Globecom Workshops (GC Wkshps), Kuala Lumpur, Malaysia, 2023, pp. 720-725, doi: 10.1109/GCWkshps58843.2023.10465014.

[6] M. Akbari, R. Bolla, R. Bruschi, F. Davoli, C. Lombardo and B. Siccardi, "A Monitoring, Observability and Analytics Framework to Improve the Sustainability of B5G Technologies," 2024 IEEE International Conference on Communications Workshops (ICC Workshops), Denver, CO, USA, 2024, pp. 969-975, doi: 10.1109/ICCWorkshops59551.2024.10615948.

[7] R. Bolla, R. Bruschi, A. Carrega, F. Davoli, C. Lombardo and B. Siccardi, “A Stochastic Knapsack Model for Energy Efficient Management of Multi-Server Queues,” 2024 7th International Balkan Conference on Communications and Networking (BalkanCom), Ljubljana, Slovenia, 2024, pp. 224-229, doi: 10.1109/BalkanCom61808.2024.10557204.

[8] R. Bolla, R. Bruschi, F. Davoli, P. Ghiorzo, L. Ivaldi, C. Lombardo and B. Siccardi, “Comparison of the Performance and Energy Efficiency Evaluation of 5G User-Plane Functions,” 2024 IEEE International Conferences on Internet of Things (iThings) and IEEE Green Computing & Communications (GreenCom) and IEEE Cyber, Physical & Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics, Copenhagen, Denmark, 2024, pp. 361-368, doi: 10.1109/iThings-GreenCom-CPSCom-SmartData-Cybermatics62450.2024.00076.

[9] M. Akbari, R. Bolla, R. Bruschi, C. Lombardo, N. S. Martinelli and B. Siccardi, “Observe to Sustain – How to Enable Beyond 5G Networks to Target Sustainability Goals,” 2024 IEEE 35th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), Valencia, Spain, 2024, pp. 1-7, doi: 10.1109/PIMRC59610.2024.10817341.

[10] R. Bolla, R. Bruschi, D. Cassinelli, F. Davoli, C. Lombardo, N. S. Martinelli and B. Siccardi, “Performance and Power Analysis of a 5G UPF Varying the C-States of the Hosting Server,” 2025 IEEE International Conference on Communications Workshops (ICC Workshops) (2025): 1025-1030.

White and non-peer reviewed papers

[11] R. Bolla, R. Bruschi, F. Davoli, C. Lombardo and B. Siccardi, “6Green: Green Technologies for 5/6G Service- Based Architectures,” 2022 IAB workshop on Environmental Impact of Internet Applications and Systems.

Participation in Research Projects

6Green, Green Technologies for 5/6G Service-Based Architectures. Funded by EU and coordinated by Prof. Roberto Bruschi (CNIT).

The 6Green project aims to conceive, design, and realize an innovative service-based and holistic ecosystem, able to extend “the communication infrastructure into a sustainable,

interconnected, greener end-to-end intercompute system" and promote energy efficiency across the whole 5/6G value-chain. The ultimate objective is to enable and to foster 5/6G networks and vertical applications reducing their carbon footprint by a factor of 10 or more.

Activity: I have designed and developed a monitoring, observability and analytics framework whose scope is power and resource consumption of a containerized B5G network (including vertical application components). This resulted in the development of a MDAF prototype which has been extensively tested. Moreover, I prepared and presented the demo at the midterm review of the project. In detail, I showcased the deployment and management (from Day 0 to Day N) of a 5G network and its live monitoring. During the demo variable traffic was generated and fed to network, meanwhile its power consumption and carbon emissions were monitored for each component of the network. I added the slices management to the above demo that was then showcased in EUCNC 2025. Finally, I took part in several 6Green meetings and in the drafting of Deliverables.

Slices-RI, Scientific LargeScale Infrastructure for Computing/Communication Experimental Studies. Funded by EU and coordinated by Prof. Serge Fdida (Sorbonne Université).

The SLICES project aims to provide researchers with a common flexible infrastructure to support research reproducibility and experiment repeatability.

Activity: my activity has concerned enhancing the testbed capabilities as far as Kubernetes and the observability framework go. Moreover, I have attended meetings and the summer schools sponsored by the project.

Attendance to workshops

IAB workshop on Environmental Impact of Internet Applications and Systems, 12 Dec. 2022, online, presenting: R. Bolla, R. Bruschi, F. Davoli, C. Lombardo, B. Siccardi, "Mobile Networks".

2023 15th International Conference on COMmunication Systems & NETworkS (COMSNETS) TASIR Workshop, 8 Jan. 2023, online, presenting: R. Bolla, R. Bruschi, C. Lombardo, S. Mangialardi, A. Mohammadpour, R. Rabbani, B. Siccardi, "A Multi-Tenant System for 5/6G Testbed as-a-Service".

Italian Networking Workshop, 16-18 Jan. 2023, Ponte di Legno, Italy, presenting: R. Bolla, R. Bruschi, F. Davoli, L. Ivaldi, C. Lombardo, B. Siccardi, “An AI Framework for Fostering 6G towards Energy Efficiency”.

Italian Networking Workshop, 22-24 Jan. 2024, Madonna di Campiglio, Italy, presenting: R. Bolla, C. Lombardo, P. Bono, N. S. Martinelli, R. Bruschi and B. Siccardi, "An Open-Source Prototype of Network Data Analytics Function for Next-Generation 5/6G Environments".

Other activities

I acted as a reviewer for the journal Computer Communications and for the 2nd workshop on green and sustainable networking (GREENNET 2023), 3rd workshop on green and sustainable networking (GREENNET 2024), 4th workshop on green and sustainable networking (GREENNET 2025) and for IEEE Networking Letters.

I co-supervised 2 master's thesis and 1 bachelor's thesis.

I was a member of the Technical Program Committee of the 3rd workshop on green and sustainable networking (GREENNET 2024) and the 4th workshop on green and sustainable networking (GREENNET 2025); both workshops are part of the IEEE ICC conference.

I prepared and presented the midterm demo of the 6Green Project in front of the consortium partners and the reviewers. The demo entailed the deployment and management (from Day 0 to Day N) of a 5G network and its live monitoring. During the demo variable traffic was generated and fed to network, meanwhile its power consumption and carbon emissions were monitored for each component of the network. Based on this demo, I also prepared the EUCNC 2025 6Green demo in which I added multiple slices to the network. Furthermore, I showed the aforementioned demo to the high school students who took part in the 2025 Unige Open Week.

Bibliography

[1] "Energy Efficiency: an Overview," https://www.gsma.com/solutions-and-impact/technologies/networks/gsma_resources/energy-efficiency-an-overview/, May 2019, accessed: 02-10-2025.

[2] "Mobile net zero state of the industry on climate action 2023," https://www.gsma.com/solutions-and-impact/connectivity-for-good/external-affairs/wp-content/uploads/2023/02/Mobile-Net-Zero-%E2%80%93-State-of-the-Industry-on-Climate-Action-2023.pdf, accessed: 02-10-2025.

[3] J. Lorincz, A. Capone, and J. Wu, "Energy-efficient and sustainable networks: State-of-the-art and new trends," *Sensors*, vol. 19, p. 4864, 2019.

[4] "ITU-R M. 2083-0," https://www.itu.int/dms_pubrec/itu-r/rec/m/R-REC-M.2083-0-201509-I!!PDF-E.pdf, 2020, accessed: 02-10-2025.

[5] G. T. Services and S. Aspects, "System Architecture for the 5G System," December 2017, online.

[6] R. Bolla, R. Bruschi, F. Davoli, and F. Cucchietti, "Energy efficiency in the future internet: A survey of existing approaches and trends in energy-aware fixed network infrastructures," *IEEE Communications Surveys Tutorials*, vol. 13, no. 2, pp. 223–244, 2011.

[7] R. Bolla, R. Bruschi, A. Carrega, F. Davoli, D. Suino, C. Vassilakis, and A. Zafeiropoulos, "Cutting the energy bills of internet service providers and telecoms through power management: An impact analysis," *Computer Networks*, vol. 56, no. 10, pp. 2320–2342, 2012, green communication networks. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128612001326

[8] "How to monitor service: The fundamental framework." https://metebalci.com/blog/a-minimum-completetutorial-of-cpu-power-management-c-states-and-p-states/, accessed: 02-10-2025.

[9] S. Beekhani, "Understanding forward and backward propagation in neural networks." https://www.linkedin.com/pulse/understanding-forward-backward-propagation-neural-suresh-beekhani-e0rkf/, March 2025, accessed: 02-10-2025.

[10] S. Mian, "Investing in the coming data revolution," https://www.nasdaq.com/articles/investing-in-the-coming-data-revolution-2019-11-06, November 2019, accessed: 02-10-2025.

[11] B. Joshi, "Breaking the energy curve: why service providers should care about 5g energy efficiency," https://www.ericsson.com/en/blog/2019/2/breaking-the-energy-curve-5g-energy-efficiency, February 2019, accessed: 02-10-2025.

[12] "Vertiv and 451 research survey on 5g energy consumtion at mwc19," https://www.vertiv.com/en-emea/about/news-and-insights/news-releases/mwc19-vertiv-and-451-research-survey-reveals-more-than-90-percent-of-operators-fear-increasing-energy-costs-for-5g-and-edge/, accessed: 02-10-2025.

[13] I. Morris, "Bill shock: Orange, china telecom fret about 5g energy costs," https://www.lightreading.com/5g/bill-shock-orange-china-telecom-fret-about-5g-energy-costs, November 2018, accessed: 02-10-2025.

[14] "Global ict energy efficiency summit paves way for 5g," https://www.mobileworldlive.com/ict-ee-18-news/global-ict-energy-efficiency-summit-paves-way-for-5g/, October 2018, accessed: 02-10-2025.

[15] European Commission, "European Green Deal," https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=celex:52019DC0640, 2019, accessed: 02-10-2025.

[16] X. Lin, "An overview of 5g advanced evolution in 3gpp release 18," *IEEE Communications Standards Magazine*, vol. 6, no. 3, pp. 77–83, 2022.

[17] D. Szabó, F. Németh, B. Sonkoly, A. Gulyás, and F. H. Fitzek, "Towards the 5G revolution: a software defined network architecture exploiting network coding as a service," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4, pp. 105–106, 2015.

[18] R. Vilalta, V. López, A. Giorgetti, S. Peng, V. Orsini, L. Velasco, R. Serral-Gracia, D. Morris, S. De Fina, F. Cugini *et al.*, "Telcofog: A unified flexible fog and

cloud computing architecture for 5G networks," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 36–43, 2017.

[19] D. Soldani and A. Manzalini, "Horizon 2020 and beyond: On the 5G operating system for a true digital society," *IEEE Vehicular Technology Magazine*, vol. 10, no. 1, pp. 32–42, 2015.

[20] "Network Functions Virtualisation - white paper," http://portal.etsi.org/NFV/NFV_White_Paper.pdf, 2012, accessed: 02-10-2025.

[21] R. Dua, A. R. Raja, and D. Kakadia, "Virtualization vs containerization to support paas," in *2014 IEEE International Conference on Cloud Engineering*. IEEE, 2014, pp. 610–614.

[22] L. Abdollahi Vayghan, M. A. Saied, M. Toeroe, and F. Khendek, "Deploying Microservice Based Applications with Kubernetes: Experiments and Lessons Learned," in *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, 2018, pp. 970–973.

[23] G. Brown, "Service-based architecture for 5g core networks," https://www.3g4g.co.uk/5G/5Gtech_6004_2017_11_Service-Based-Architecture-for-5G-Core-Networks_HR_Huawei.pdf, 2017, accessed: 02-10-2025.

[24] "5g system architecture," https://www.etsi.org/deliver/etsi_ts/123500_123599/123501/15.03.00_60/ts_123501v150300p.pdf, accessed: 02-10-2025.

[25] "Vertiv and 451 research survey on 5g energy consumtion at mwc19," https://www.vertiv.com/en-emea/about/news-and-insights/news-releases/mwc19-vertiv-and-451-research-survey-reveals-more-than-90-percent-of-operators-fear-increasing-energy-costs-for-5g-and-edge/, accessed: 02-10-2025.

[26] A. Khichane, I. Fajjari, N. Aitsaadi, and M. Gueroui, "5gc-observer demonstrator: a non-intrusive observability prototype for cloud native 5g system," in *NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium*, 2023, pp. 1–3.

[27] W. John, F. Moradi, B. Pechenot, and P. Sköldström, "Meeting the observability challenges for vnfs in 5g systems," in *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, 2017, pp. 1127–1130.

[28] V. Stoynov, V. Poulkov, Z. Valkova-Jarvis, G. Iliev, and P. Koleva, "Ultra-dense networks: Taxonomy and key performance indicators," *Symmetry*, vol. 15, no. 1, 2023. [Online]. Available: https://www.mdpi.com/2073-8994/15/1/2

[29] J. Lorincz, A. Capone, and J. Wu, "Greener, energy-efficient and sustainable networks: State-of-the-art and new trends," *Sensors*, vol. 19, no. 22, 2019. [Online]. Available: https://www.mdpi.com/1424-8220/19/22/4864

[30] "Huawei first 5g power solution," https://www.huawei.com/en/pressevents/news/2018/10/Huawei-First-5G-Power-Solution, October 2010, accessed: 02-10-2025.

[31] F. T. C. R. Danilak, "Why Energy Is A Big And Rapidly Growing Problem For Data Centers," https://www.forbes.com/sites/forbestechcouncil/2017/12/15/why-energy-is-a-big-and-rapidly-growing-problem-for-data-centers/, December 2017, accessed: 02-10-2025.

[32] J. Malmodin, "The power consumption of mobile and fixed network data services-the case of streaming video and downloading large files," in *Electronics Goes Green*, vol. 2020, 2020, p. 12.

[33] "Mobile net zero state of the industry on climate action," https://www.gsma.com/solutions-and-impact/connectivity-for-good/external-affairs/wp-content/uploads/2025/07/The_GSMA-Mobile-Net-Zero-2025-State-of-the-Industry-on-Climate-Action.pdf, 2025, accessed: 02-10-2025.

[34] G. Liu, Y. Huang, N. Li, J. Dong, J. Jin, Q. Wang, and N. Li, "Vision, requirements and network architecture of 6g mobile network beyond 2030," *China Communications*, vol. 17, no. 9, pp. 92–104, 2020.

[35] X. Zhang, Z. Shen, B. Xia, Z. Liu, and Y. Li, "Estimating power consumption of containers and virtual machines in data centers," in *2020 IEEE International Conference on Cluster Computing (CLUSTER)*, 2020, pp. 288–293.

[36] K. N. Khan, M. Hirki, T. Niemi, J. K. Nurminen, and Z. Ou, "Rapl in action: Experiences in using rapl for power measurements," *ACM Trans. Model. Perform. Eval. Comput. Syst.*, vol. 3, no. 2, Mar. 2018. [Online]. Available: https://doi.org/10.1145/3177754

[37] R. Bolla, R. Bruschi, A. Gallo, C. Lombardo, and N. S. Martinelli, "To scale or not to scale? understand the overhead of container scaling operations," in *2025 21st International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT)*, 2025, pp. 890–894.

[38] R. Bolla, R. Bruschi, F. Davoli, C. Lombardo, J. F. Pajo, and B. Siccardi, "Machine-learning-based 5g network function scaling via black- and white-box kpis," in *2023 21st Mediterranean Communication and Computer Networking Conference (MedComNet)*, 2023, pp. 143–150.

[39] "Advanced configuration and power interface (acpi) specification," https://uefi.org/sites/default/files/resources/ACPI_6_2.pdf, May 2017, accessed: 02-10-2025.

[40] "6Green - Green Technologies for 5/6G Service-Based Architectures," https://www.6green.eu/, accessed: 02-10-2025.

[41] R. Bolla, R. Bruschi, C. Lombardo, and B. Siccardi, "6g enablers for zero-carbon network slices and vertical edge services," *IEEE Networking Letters*, vol. 5, no. 3, pp. 173–176, 2023.

[42] M. Jimeno, K. Christensen, and B. Nordman, "A network connection proxy to enable hosts to sleep and save energy," in *2008 IEEE International Performance, Computing and Communications Conference*, 2008, pp. 101–110.

[43] R. Bolla, M. Giribaldi, R. Khan, and M. Repetto, "Network connectivity proxy: Architecture, implementation, and performance analysis," *IEEE Systems Journal*, vol. 11, no. 2, pp. 588–599, 2017.

[44] G. A. Carella, M. Pauls, L. Grebe, and T. Magedanz, "An extensible autoscaling engine (ae) for software-based network functions," in *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, 2016, pp. 219–225.

[45] A. Beloglazov and R. Buyya, "Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers," in *Proceedings of the 8th International Workshop on Middleware for Grids, Clouds and e-Science*, ser. MGC '10. New York, NY, USA: Association for Computing Machinery, 2010. [Online]. Available: https://doi.org/10.1145/1890799.1890803

[46] S. Dutta, T. Taleb, and A. Ksentini, "Qoe-aware elasticity support in cloud-native 5g systems," in *2016 IEEE International Conference on Communications (ICC)*, 2016, pp. 1–6.

[47] "Kubernetes," https://kubernetes.io/docs/concepts/overview, accessed: 02-10-2025.

[48] W. Shu and Y. Li, "Joint offloading strategy based on quantum particle swarm optimization for mec-enabled vehicular networks," *Digital Communications and Networks*, vol. 9, no. 1, pp. 56–66, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S235286482200030X

[49] "A minimum complete tutorial of cpu power management, c-states and p-states." https://metebalci.com/blog/a-minimum-complete-tutorial-of-cpu-power-management-c-states-and-p-states/, accessed: 02-10-2025.

[50] M. Usman, S. Ferlin, A. Brunstrom, and J. Taheri, "A survey on observability of distributed edge container-based microservices," *IEEE Access*, vol. 10, pp. 86 904–86 919, 2022.

[51] IBM, "The enterprise guide to observability,," https://www.ibm.com/resources/automate/observability/, accessed: 02-10-2025.

[52] "cAdvisor," https://github.com/google/cadvisor, accessed: 02-10-2025.

[53] "Kube State Metrics," https://github.com/kubernetes/kube-state-metrics, accessed: 02-10-2025.

[54] "Kepler," https://github.com/sustainable-computing-io/kepler, accessed: 02-10-2025.

[55] "Scaphandre," https://github.com/hubblo-org/scaphandre, accessed: 02-10-2025.

[56] "Raritan PDU exporter," https://github.com/psyinfra/prometheus-raritan-pdu-exporter, accessed: 02-10-2025.

[57] "Node Exporter," https://github.com/prometheus/node_exporter, accessed: 02-10-2025.

[58] "Prometheus," https://prometheus.io/, accessed: 02-10-2025.

[59] P. Armitage and T. Colton, *Encyclopedia of biostatics*, 2005.

[60] T. Wang, X. Dai, and Y. Liu, "Learning with hilbert–schmidt independence criterion: A review and new perspectives," *Knowledge-Based Systems*, vol. 234, p. 107567, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705121008297

[61] W. Huo, W. Wang, and W. Li, "Anomalydetect: An online distance-based anomaly detection algorithm," in *Web Services – ICWS 2019*, J. Miller, E. Stroulia, K. Lee, and L.-J. Zhang, Eds. Cham: Springer International Publishing, 2019, pp. 63–79.

[62] C. Zhang, W. Zuo, P. Yang, Y. Li, and X. Wang, "Outsourced privacy-preserving anomaly detection in time series of multi-party," *China Communications*, vol. 19, no. 2, pp. 201–213, 2022.

[63] K. Chen, "Indirect pca dimensionality reduction based machine learning algorithms for power system transient stability assessment," in *2019 IEEE Innovative Smart Grid Technologies - Asia (ISGT Asia)*, 2019, pp. 4175–4179.

[64] B. Guenin, J. Könemann, and J. Tunçel, *A Gentle Introduction to Optimization*, 2014.

[65] K. Pearson, *Notes on regression and inheritance in the case of two parents*, 1895, vol. 58.

[66] A. Gretton, O. Bousquet, A. Smola, and B. Scholkopf, "Measuring statistical dependence with hilbert-schmidt norms," in *Algorithmic Learning Theory*, S. Jain, H. U. Simon, and E. Tomita, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 63–77.

[67] I. Steinwart, "On the influence of the kernel on the consistency of support vector machines," *J. Mach. Learn. Res.*, vol. 2, p. 67–93, Mar. 2002. [Online]. Available: https://doi.org/10.1162/153244302760185252

[68] "Iperf," https://www.iperf.fr/, accessed: 02-10-2025.

[69] R. Bruschi, R. Bolla, F. Davoli, A. Zafeiropoulos, and P. Gouvas, "Mobile edge vertical computing over 5g network sliced infrastructures: An insight into integration approaches," *IEEE Communications Magazine*, vol. 57, no. 7, pp. 78–84, 2019.

[70] M. Akbari, R. Bolla, R. Bruschi, C. Lombardo, N. S. Martinelli, and B. Siccardi, "Observe to sustain – how to enable beyond 5g networks to target sustainability goals," in *2024 IEEE 35th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, 2024, pp. 1–7.

[71] C. Hsiung, F. J. Lin, J.-C. Chen, and C. Chen, "5g network slice scalability based on management data analytics function (mdaf)," in *New Trends in Computer Technologies and Applications*, S.-Y. Hsieh, L.-J. Hung, R. Klasing, C.-W. Lee, and S.-L. Peng, Eds. Singapore: Springer Nature Singapore, 2022, pp. 587–598.

[72] E. Pateromichelakis, F. Moggio, C. Mannweiler, P. Arnold, M. Shariat, M. Einhaus, Q. Wei, Bulakci, and A. De Domenico, "End-to-end data analytics framework for 5g architecture," *IEEE Access*, vol. 7, pp. 40 295–40 312, 2019.

[73] H. Zafar, U. Fattore, F. Cirillo, and C. J. Bernardos, "Data usage control for privacy-enhanced network analytics in private 5g networks," *IEEE Open Journal of the Communications Society*, pp. 1–1, 2024.

[74] D. Bega, M. Gramaglia, R. Perez, M. Fiore, A. Banchs, and X. Costa-Pérez, "Ai-based autonomous control, management, and orchestration in 5g: From standards to algorithms," *IEEE Network*, vol. 34, no. 6, pp. 14–20, 2020.

[75] "5G; Management and orchestration; Concepts, use cases and requirements (3GPP TS 28.530 version 16.2.0 Release 16)," available at https://www.etsi.org/deliver/etsi_ts/128500_128599/128530/16.02.00_60/ts_128530v160200p.pdf, accessed: 02-10-2025.

[76] A. Bellin, F. Granelli, and D. Munaretto, "A measurement-based approach to analyze the power consumption of the softwarized 5g core," *Computer Networks*, vol. 244, p. 110312, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128624001440

[77] R. Bolla, P. Bono, R. Bruschi, C. Lombardo, N. S. Martinelli, and B. Siccardi, "An open-source prototype of network data analytics function for next-generation 5/6g environments," in *2023 IEEE Globecom Workshops (GC Wkshps), Kuala Lumpur, Malaysia*, 2023, pp. 720–725.

[78] "prometheus-api-client," https://pypi.org/project/prometheus-api-client/, accessed: 02-10-2025.

[79] " OPENAIR-CN-5G: An implementation of the 5G Core network by the OpenAirInterface community," available at https://gitlab.eurecom.fr/oai/cn5g/oai-cn5g-fed/-/tree/master, accessed: 02-10-2025.

[80] "LoadCore - core-network-solutions," available at https://www.keysight.com/it/en/product/P8900S/loadcore-corenetwork-solutions.html., accessed: 02-10-2025.

[81] S. Nedevschi, L. Popa, G. Iannaccone, S. Ratnasamy, and D. Wetherall, "Reducing network energy consumption via sleeping and rate-adaptation," in *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, ser. NSDI'08. USA: USENIX Association, 2008, p. 323–336.

[82] R. Bolla, R. Bruschi, and P. Lago, "The hidden cost of network low power idle," in *2013 IEEE International Conference on Communications (ICC)*, 2013, pp. 4148–4153.

[83] J. Rischke, C. Vielhaus, P. Sossalla, J. Wang, and F. H. Fitzek, "Comparison of upf acceleration technologies and their tail-latency for urllc," in *2022 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, 2022, pp. 19–25.

[84] T. A. N. do Amaral, R. V. Rosa, D. F. C. Moura, and C. E. Rothenberg, "An in-kernel solution based on xdp for 5g upf: Design, prototype and performance evaluation," in *2021 17th International Conference on Network and Service Management (CNSM)*, 2021, pp. 146–152.

[85] F. Paolucci, D. Scano, F. Cugini, A. Sgambelluri, L. Valcarenghi, C. Cavazzoni, G. Ferraris, and P. Castoldi, "User plane function offloading in p4 switches for enhanced 5g mobile edge computing," in *2021 17th International Conference on the Design of Reliable Communication Networks (DRCN)*, 2021, pp. 1–3.

[86] "Multus." https://github.com/k8snetworkplumbingwg/multus-cni/, accessed: 02-10-2025.

[87] M. Barbosa, M. Silva, E. Cavalcanti, and K. Dias, "Open-source 5g core platforms: A low-cost solution and performance evaluation," in *2025 International Conference on Information Networking (ICOIN)*, 2025, pp. 99–104.

[88] W. Rankothge, H. Ramalhinho, and J. Lobo, "On the scaling of virtualized network functions," in *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, 2019, pp. 125–133.

[89] F. B. Carvalho, R. A. Ferreira, Cunha, M. A. M. Vieira, and M. K. Ramanathan, "Dyssect: Dynamic scaling of stateful network functions," in *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*, 2022, pp. 1529–1538.

[90] "Zero touch network and service management (zsm)." https://www.etsi.org/committee/eni?tmpl=component, accessed: 02-10-2025.

[91] "Etsi experiential networked intelligence (eni)." https://www.etsi.org/technologies/zero-touch-network-servicemanagement, accessed: 02-10-2025.

[92] A. M. Marchevsky, "Chapter 9 - the use of artificial neural networks for the diagnosis and estimation of prognosis in cancer patients," in *Outcome Prediction in Cancer*, A. F. Taktak and A. C. Fisher, Eds. Amsterdam: Elsevier, 2007, pp. 243–259. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780444528551500118

[93] A. A. Pour Haji Kazem, "Chapter 1 - neural networks," in *Decision-Making Models*, ser. Uncertainty, Computational Techniques, and Decision Intelligence, T. Allahviranloo, W. Pedrycz, and A. Seyyedabbasi, Eds. Academic Press, 2024, pp. 3–8. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780443161476000177

[94] T. Subramanya and R. Riggio, "Machine learning-driven scaling and placement of virtual network functions at the network edges," in *2019 IEEE Conference on Network Softwarization (NetSoft)*, 2019, pp. 414–422.

[95] G. Kousiouris, A. Menychtas, D. Kyriazis, K. Konstanteli, S. V. Gogouvitis, G. Katsaros, and T. A. Varvarigou, "Parametric design and performance analysis of a decoupled service-oriented prediction framework based on embedded numerical software," *IEEE Transactions on Services Computing*, vol. 6, no. 4, pp. 511–524, 2013.

[96] X. Zhang, T. Xue, and H. Eugene Stanley, "Comparison of econometric models and artificial neural networks algorithms for the prediction of baltic dry index," *IEEE Access*, vol. 7, pp. 1647–1657, 2019.

[97] "Scikit-learn." https://scikit-learn.org/, accessed: 02-10-2025.

[98] "Tensorflow." https://www.tensorflow.org/, accessed: 02-10-2025.

[99] "Keras." https://keras.io/, accessed: 02-10-2025.

[100] "Pytorch." https://pytorch.org/, accessed: 02-10-2025.

[101] C. Bishop, *"Pattern Recognition and Machine Learning,"*. Springer, January 2006.

[102] A. Burkov, *"The hundred-page machine learning book,"*, 2019.

[103] "Optimizer that implements the adam algorithm." https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/Adam, accessed: 02-10-2025.

[104] "Ueransim: Open source 5G UE and RAN (gNodeb) implementation," https://github.com/aligungr/UERANSIM, accessed: 02-10-2025.

[105] "Free5gc." https://www.free5gc.org/, accessed: 02-10-2025.

[106] "The sequential model." https://www.tensorflow.org/guide/keras/sequential_model, accessed: 02-10-2025.

[107] "Densely-connected nn layers." https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dense, accessed: 02-10-2025.

Power Measurements and Saving Techniques in Beyond 5G Networks

Beatrice Siccardi