Tesis Doctoral por Compendio de Publicaciones

Energy-aware function and resource management in next-gen networks with variability models

UNIVERSIDAD DE MÁLAGA

Ángel Jesús Cañete Valverde

Programa de Doctorado en Tecnologías Informáticas
Departamento de Lenguajes y Ciencias de la Computación

Universidad de Málaga

Supervised by

Prof. Dr. Lidia Fuentes Fernández
Dr. Mercedes Amor Pinilla

January 2024

AUTOR: Ángel Jesús Cañete Valverde

https://orcid.org/0000-0001-8422-6063

EDITA: Publicaciones y Divulgación Científica. Universidad de Málaga

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD DE LA TESIS PRESENTADA PARA OBTENER EL TÍTULO DE DOCTOR

D./Dña ÁNGEL JESÚS CAÑETE VALVERDE
Estudiante del programa de doctorado EN TECNOLOGÍAS INFORMÁTICAS de la Universidad de Málaga, autor/a de la tesis, presentada para la obtención del título de doctor por la Universidad de Málaga, titulada: ENERGY-AWARE FUNCTION AND RESOURCE MANAGEMENT IN NEXT-GEN NETWORKS WITH VARIABILITY MODELS
Realizada bajo la tutorización de LIDIA FUENTES FERNÁNDEZ y dirección de LIDIA FUENTES FERNÁNDEZ Y MARÍA MERCEDES AMOR PINILLA

DECLARO QUE:

La tesis presentada es una obra original que no infringe los derechos de propiedad intelectual ni los derechos de propiedad industrial u otros, conforme al ordenamiento jurídico vigente (Real Decreto Legislativo 1/1996, de 12 de abril, por el que se aprueba el texto refundido de la Ley de Propiedad Intelectual, regularizando, aclarando y armonizando las disposiciones legales vigentes sobre la materia), modificado por la Ley 2/2019, de 1 de marzo.

Igualmente asumo, ante a la Universidad de Málaga y ante cualquier otra instancia, la responsabilidad que pudiera derivarse en caso de plagio de contenidos en la tesis presentada, conforme al ordenamiento jurídico vigente ya que nada ha sido utilizado en tesis anteriores.

En Málaga, a 09 de ENERO de 2024

Fdo.: ÁNGEL JESÚS CAÑETE VALVERDE Doctorando/a	Fdo.: LIDIA FUENTES FERNÁNDEZ Tutor/a
Fdo.: LIDIA FUENTES FERNÁNDEZ Director/es de tesis	MARÍA MERCEDES AMOR PINILLA

UNIVERSIDAD DE MÁLAGA
DEPARTAMENTO DE LENGUAJES Y CIENCIAS DE LA COMPUTACIÓN

La Prof. Dra. Doña Lidia Fuentes Fernández, Catedrática de Universidad y la Dra. Doña María Mercedes Amor Pinilla, Titular de Universidad, ambas pertenecientes al Área de Ingeniería Telemática, de la E.T.S. de Ingeniería Informática de la Universidad de Málaga,

certifican que Don Ángel Jesús Cañete Valverde, graduado y máster en Ingeniería Informática, ha realizado en el Departamento de Lenguajes y Ciencias de la Computación de la Universidad de Málaga, bajo nuestra dirección, el trabajo de investigación correspondiente a su Tesis Doctoral titulado:

Energy-aware function and resource management in next-gen networks with variability models

En ella se han propuesto aportaciones originales a la ingeniería del software y los resultados han dado lugar a las publicaciones indexadas que avalan esta "Tesis por compendio de publicaciones" y que no han sido utilizadas en tesis anteriores:

1. Energy-efficient adaptation engines for android applications
2. Energy-efficient deployment of IoT applications in edge-based infrastructures. A software product line approach
3. Supporting IoT applications deployment on edge-based infrastructures using multi-layer feature models
4. HADES: An NFV solution for energy-efficient placement and resource allocation in heterogeneous infrastructures

Por todo ello, consideran que esta Tesis es apta para su presentación al Tribunal que ha de juzgarla. Y para que conste a efectos de lo establecido, se AUTORIZA la presentación de esta Tesis en la Universidad de Málaga.

Málaga, enero de 2024

Fdo.: Lidia Fuentes Fernández Fdo.: María Mercedes Amor Pinilla

Acknowledgements

This PhD. thesis has been supported by the European Union's H2020 research and innovation programme under grant agreement DAEMON H2020-101017109, by the projects IRIS PID2021-12281 2OB-I00 (co-financed by FEDER funds) and LEIA UMA18-FEDERIA-157.

Special Acknowledgements

Y este camino, se termina. Mentiría si dijese que ha sido fácil. Más allá de las complicaciones propias de realizar una tesis, la pandemia que sacudió el planeta y que nos dejó aislados hizo que este proceso de dilatara y me bloqueó esa inspiración tan necesaria para investigar. No he caminado solo, sino acompañado por unas personas a las que quiero mostrar mi agradecimiento.

A mis supervisoras, por apostar por mí y descubrirme la senda de la investigación.

A mis compañeros del laboratorio 3.3.3, por los buenos momentos y brindarme su ayuda cuando la he necesitado.

A mi familia, por estar ahí siempre. Mención especial a mis padres, por su lucha para poder procurarme una formación académica que ellos no tuvieron la suerte de disfrutar. Gracias.

A mis amigos, a los de siempre y a los que se han ido sumando. Especialmente a Jesús, mi hermano de otra madre, con quien he compartido almuerzos durante el desarrollo de esta tesis que han sido agua en el camino.

Y como no a Fátima, mi chica y mi suerte, quien me acompaña prácticamente desde que inicié esta tesis. Por su apoyo incondicional y su comprensión. Por confiar en mi cuando a mi me cuesta. Por estar y por ser.

Esto no es un final, sino un comienzo.

Contents

List of Figures

List of Tables

Part I

Thesis

Chapter 1

Introduction

In the current world, there is a growing emphasis on adopting sustainable practices in every process and activity. The primary goal is to mitigate global warming and address the energy production crisis. With this aim, major Information Technology (**IT**) providers, as well as network operators and telecommunication companies, are actively promoting greener and more sustainable networking solutions. These efforts focus on two main aspects: reducing energy consumption and prioritizing the utilization of infrastructure powered by renewable energy sources. Then, the forthcoming generations of mobile networks, Beyond 5G (**B5G**) are dedicated to achieving energy efficiency while simultaneously promising improved capacity, lower latency, and enhanced reliability. The capabilities of 5G/B5G technology facilitate faster communication among a diverse range of connected devices, thereby fueling the expansion and commercial prosperity of Internet of Things (**IoT**) systems.

At the same time, and fostered by the capabilities of 5G/B5G technology, the IoT and *cyber-physical systems* (**CPSs**) are experiencing an exceptional surge. Such systems generate a large amount of data produced by a myriad of devices, ranging from smartphones and sensors to home appliances and all kinds of wearables. These devices usually have limited computational capacities, making it impossible for them to process the data they generate. Therefore, they require sending the produced data to high-performance computing resources capable of processing them. The current enhanced communication capabilities driven by 5G/B5G are promoting a revolutionary transformation in various sectors by introducing novel services and applications. These sectors include environmental monitoring, mobile health systems, intelligent transportation systems, and so on [1].

According to the Royal Society, information technologies are responsible for approximately 1.4% to 5.9% of worldwide greenhouse gas emissions. How software is designed, developed, and deployed plays a crucial role in determining its

energy consumption. The software itself does not directly consume energy or generate harmful emissions. Indeed, the IT energy consumption issue lies in the development practices employed and the subsequent utilization of resources by the underlying software. The software is responsible for managing hardware resources, and therefore, indirectly affects the energy consumption. At the same time, since software relies on hardware to be executed, the increasing scale of software usage corresponds to a heightened reliance on machines to support its execution.

Until recently, Cloud Computing has posed as the dominant paradigm for offering a centralized solution that can efficiently handle vast data storage requirements and meet the demanding computational needs for processing data from IoT devices [2]. The large capacity of cloud data centres has allowed this model to work correctly during the last years but is paying a heavy cost in terms of energy and latency. As expected, the massive volume of data that needs to be transmitted to the cloud has the potential to congest networks, leading to boosted latency and increased energy consumption [3]. On the one hand, the growing demands to reduce the energy consumption of cloud centres, and on the other the increasing interest of CPSs to offer a better quality of service (**QoS**), has led to the emergence of new deployment paradigms as an alternative to Cloud Computing.

Edge Computing (**EC**) [4, 5, 3] has become both an alternative and complementary solution to the computing services offered by the cloud by shifting the provision of computing services from the cloud, in the core of the Internet, to the edge of the Internet, proposing a more sustainable solution. The EC paradigm allows to take advantage of the inactive computational capacity and unused storage space of edge devices placed at the Internet's frontier (e.g., routers or switches). This means that part of the data processing and storage will be displaced to this heterogeneous set of devices closer to where data and services are produced and consumed, reducing the data transmitted through the Internet. The combined use of Edge and Cloud infrastructures is a promising solution to reduce the energy footprint of IoT/B5G services while maintaining the required QoS, reducing latency and alleviating the network workload [6].

Despite its advantages, the realization of edge-based deployments is not, in practical terms, an easy issue [4, 7, 6, 8] due to the heterogeneity of devices, resources and technologies used in these infrastructures. The solution to achieve long-sought sustainability requires the amalgamation and seamless integration of innovative technologies and solutions sourced from diverse domains.

As environmental sustainability has become a pivotal focus in the development of future telecommunication systems [9] and distributed services, and to mitigate the environmental impact caused by the growing number of IoT devices and combat global warming, international initiatives are advocating for resource-efficient solutions at different levels.

From a holistic perspective, having sustainable systems begins with software development processes, in which being aware of the infrastructures where software functions will be executed is increasingly gaining importance. With the aim of optimally managing distributed infrastructure resources, complex functionalities such as IoT and network services need to be broken down into a series of tasks or functions. However, the identification and separation of tasks are usually done manually and for a specific application, ignoring the characteristics of the host devices [7, 10, 11, 5].

In order to attain efficient deployment, it is imperative for the application to be aware of the target host devices and their characteristics, as it becomes essential to determine how these tasks can be fine-tuned and then distributed within a heterogeneous infrastructure encompassing B5G, Edge, and Cloud environments. Regarding functional requirements, certain software components may need specific software and hardware infrastructure elements or features (e.g., a video camera), thereby making it infeasible to execute certain software functions on a given infrastructure. In addition, in order to ensure that the non-functional requirements (e.g., a certain QoS) are fulfilled, it is imperative to have knowledge prior to execution not only of the components presented in the edge infrastructure (e.g., a specific network card) but also of the characteristics associated with these components (e.g., its interface and performance capabilities). This awareness and understanding enables informed decision-making regarding resource allocation, performance optimization, and effective management of the infrastructure at the deployment time. However, it could hamper its applicability, especially if we bring energy consumption into the equation.

To alleviate the complexity and burden of this intricate and potentially unfamiliar process for developers, such task separation and the earlier incorporation of the infrastructure concerns can be integrated into software production through *Software Product Lines* (**SPL**). SPL is a systematic approach to developing a variety of related software products that share a common set of core features and components. The main activity of SPLs is variability modelling, in which the common elements and variabilities of the system are specified. Despite its significance, infrastructure considerations are frequently overlooked in SPL models. When they are included, the modelling of infrastructure-specific features becomes intertwined with other application-specific features, resulting in a one-to-one correspondence between the application's features and software components. By incorporating infrastructure considerations in the deployment of IoT systems, it becomes possible to prevent the creation of unsupported software, optimize the software to the infrastructure making it more energy efficient, assist users during infrastructure acquisition, and optimize resource utilization during execution.

After software development and configuration, the next step is to assign tasks to

the devices. This decision should consider the computational and communication latency and expected energy consumption, and find different task assignment solutions depending on their resource demand and delay sensitivity [7, 6, 8]. Within these available solutions, in this thesis we focus on those that minimize energy consumption, thus promoting the development of more environmentally sustainable applications while minimizing latency to ensure a good quality of user experience. Considering that infrastructure is shared among multiple applications, the goal is to identify subsets of devices that can execute each task with their available resources. The objective is to select the most appropriate devices to minimize energy consumption and/or latency. Knowing the hardware and software information (characteristics and available resources) of host devices, the assignment can be done with an energy-aware scheduling policy that assures QoS.

Then, the energy-aware scheduling solution should be integrated with a resource orchestration platform that performs the deployment. Orchestration platforms automate various tasks, simplifying deployment, configuration, and management of edge/cloud devices and services. This automation reduces manual effort and provides centralized control, making it easier to monitor and maintain the edge computing infrastructure. Integrating our proposal with an orchestration platform, facilitates its adoption and the validation of how effective is the energy-aware scheduling solution proposed.

Finally, when applied to mobile scenarios like Mobile Edge Computing (**MEC**), part of the application functionality will be deployed on edge/cloud devices (this process is known as task-offloading). What functionalities are delegated to other devices beyond the user device will depend on the context in the broader sense, considering both the user's own device (e.g., bandwidth, battery) and the edge/-cloud devices (available resources, hardware and software characteristics, etc). The user's behaviour plays a key role in this context, as it can be leveraged to reduce energy consumption. For instance, if the user does not use certain features of an application, the application can be modified to not activate those features from the beginning, thereby reducing the number of tasks that need to be offloaded or executed on the user's device. Such an adaption process must be energy efficient and transparent to the user, to minimize energy impact and preserve the quality of the user experience.

In this thesis, we address the aforementioned research challenges providing a full-stack solution framework that spans from application variability characterization to tackle an energy-aware deployment of software functions (e.g., network functions) considering the current availability of software and hardware resources inside a heterogeneous infrastructure. Specifically, we have conducted: a) a set of modules to assist distributed software development in heterogeneous edge infrastructures using SPL [12]; b) an energy-efficient task assignment framework for

IoT/Edge/Cloud/B5G infrastructures [13]; c) a framework that works as an extension of the European Telecommunication Standard Organization (**ETSI**) hosted project Open Source MANO for energy-efficient creation, deployment, modification and orchestration of workloads on edge infrastructures [14]; and d) a set of energy-efficient software adaptation engines for smartphones [15].

Chapter 2

Background

This section presents the background necessary to understand the work presented in this thesis. It provides a concise overview of the existing knowledge on this topic and outlines the contribution our research will make to the current understanding. Specifically, it delineates the fundamental principles of Software Product Lines and variability models while introducing the core concepts of Edge Computing.

2.1 Software Product Lines

The baseline software engineering technologies used in this work are software product lines and variability models. This section provides a brief introduction of the main concepts that characterized these technologies.

2.1.1 Base concepts

A *Software Product Line* is defined as *"a set of software-intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way"* [16]. A *feature* is a characteristic or behaviour visible to the end user of a software system. Features are used in SPL engineering to specify and document product commonalities and differences, and to guide structure, reuse and variation in all phases of the software life cycle. A specific product is identified by a subset of features, called *configuration*. Since not all configurations are valid and result in meaningful products, there are constraints that are explicitly modelled in SPLs, called *dependencies*.

The classical SPL engineering paradigm separates two main processes: *Domain Engineering* and *Application Engineering*. Domain engineering is the process of analyzing the knowledge area (i.e., domain) of an SPL and developing reusable

artefacts. On the other hand, application engineering is the process of developing a specific product for a specific customer's needs. The specific characteristics of SPLs also lead to a separation between the problem space and the solution space. The problem space takes the stakeholder perspective and their problems, requirements, and points of view of the entire domain and of the individual products. Features are abstract representations specific to the domain, defining the characteristics of the problem space. On the other hand, the solution space pertains to the perspectives of developers and suppliers, encompassing terminology and activities related to feature design, implementation, validation, and verification. The goal of the solution space is to facilitate systematic reuse by enabling the straightforward combination and utilization of features.

2.1.2 Variability modelling

The main activity of SPLs is variability modelling, in which the common elements and variabilities of the system are specified and that belongs to the Domain Engineering process [17] using feature models (**FMs**). In terms of features, an FM represents which elements of a product family (either a family of applications or a family of systems) are common, which are variable and the relationship between them. FMs are represented as a set of hierarchically ordered features, composed of parent-child relationships and a set of constraints (called *cross-tree constraints*) which represent the relationships among them. Features can be optional or mandatory and have groups of features as alternatives. It was first introduced by Kang as part of the Feature-Oriented Domain Analysis in [17].

A feature model captures the variability of the application domain of an SPL. Nevertheless, an SPL may also contain domain-independent variability, such as external features [18]. Examples include security requirements, execution environment, and the runtime context (e.g., the battery level of a mobile device, its location, etc). In addition, the SPL may have internal variability, such as the implementation variability offered by components. An SPL product must also satisfy the requirements of these additional *variability dimensions*. This is achieved through the use of several feature models (called *multi-layer FMs*), each of which models one SPL's dimension of variability [19]. To ensure consistency across FMs, *cross-model constraints* are established to govern the relationships between models [20, 21].

Despite its importance, infrastructure considerations are often neglected in SPL models. When considered, the modelling of infrastructure-specific features tends to intertwine with application-specific features, leading to a direct mapping between the application's features and software components. By integrating infrastructure considerations into the deployment of IoT systems, it becomes feasible to avoid the creation of unsupported software, optimize software for the infrastructure to

enhance energy efficiency, assist users in infrastructure acquisition, and optimize resource utilization during execution [12].

2.1.3 Solvers: Enabling Modelling and Automated Reasoning

A solver is software tool designed to solve a specific type of problem. Its main purpose is to automate the process of finding solutions to complex problems that may be difficult or time-consuming to solve manually. Solvers can handle large-scale problems, explore solution spaces, evaluate possibilities, and provide optimal or near-optimal solutions based on defined objectives and constraints.

Solvers play a crucial role in variability modeling by enabling efficient analysis, reasoning, and derivation of valid configurations. By means of constraint satisfaction methods, solvers help to determine the solution space of the given variability model. Techniques such as backtracking, constraint propagation, and local search are commonly employed to tackle the constraint satisfaction problems inherent in variability modeling.

Solvers vary in terms of their capabilities, performance, and the types of problems they can solve. The most popular types of solvers applied to variability modelling are *sat*isfiability (**SAT**), *Satisfiability Modulo Theories* (**SMT**) and *Constraint Programming* (**CP**). SAT solvers address the Boolean satisfiability problem, which involves finding a truth assignment for Boolean variables that satisfies a given propositional formula. SAT solvers are particularly useful as they can handle classical FMs by converting them into propositional formulas, allowing them to be effectively solved using SAT solvers [22]. SMT solvers extend the scope of problem-solving beyond the Boolean satisfiability problem by incorporating more intricate formulas that involve numerical variables, arithmetic operations, and diverse data structures like lists, arrays, and bit vectors. In practical applications, SMT solvers combine the capabilities of SAT solvers with additional theory solvers [23]. SMT solvers are capable of handling not only propositional formulas but also non-Boolean formulas and arithmetic expressions. Finally, CP solvers offer a versatile approach to problem-solving by tackling Boolean, numerical, and linear problems through techniques such as backtracking and constraint propagation. These solvers effectively reduce variable domains, strengthen existing constraints, and even introduce new constraints to find solutions [24]. Additionally, CP solvers have the capability to handle non-Boolean formulas and arithmetic expressions, making them a comprehensive tool for a wide range of problem domains.

The performance and efficiency of solvers can vary depending on its type and the complexity of the problem they are designed to solve. In general terms, solvers that support a wider range of variables and operations tend to have lower perfor-

mance in terms of speed and memory usage.

2.2 Energy consumption in IT

As organizations strive to optimize their operations, reduce costs, and minimize their environmental impact, the role of software efficiency has emerged as a critical factor. While software systems themselves do not directly consume energy, they have a significant impact on hardware utilization, leading to indirect energy consumption. This becomes particularly critical in edge computing systems, such as CPSs, which are powered by smart devices like smartwatches that often face limitations due to the scarcity of lithium batteries.

In order to reduce the energy consumption of CPSs, a comprehensive examination of the various actors, components, and the surrounding environment is necessary. It is important to recognize that energy consumption extends beyond the device itself, encompassing the entire ecosystem affected by CPS usage, including networking, servers, data centers, and database storage, among others. User behavior when interacting with applications also plays a pivotal role in the energy consumption of software. Actions such as excessive multitasking, running resource-intensive applications simultaneously, or leaving applications running in the background can significantly impact energy usage. Adapting applications to deactivate certain parts of their execution based on user behavior can effectively reduce the energy consumption.

2.3 Computing virtualization

Virtualization is a computing technique that uses software to mimic the characteristics of hardware to create a virtual computer system. This concept emerged in the mid-1960s [25] when IBM required running multiple tasks on a mainframe in parallel. Virtualization enables the concurrent execution of multiple virtual machines (**VMs**), allowing the hosting of multiple operating systems (**OS**) and applications on a single hardware server.

Resource virtualization is a key technology in edge and cloud computing to decouple hardware resources from software and run multiple tenants on the same hardware [26]. However, EC has significant differences in terms of location awareness, mobility-based services, heterogeneous and resource-constrained devices, and wide distribution of devices. These differences limit the use of hardware-level virtualization (based on VMs), that runs the *virtual machine monitor* (also called *hipervisor*) directly on either hardware or OS, in edge computing scenarios. Such constraints lead to using lightweight virtualization techniques based on containers

[27] and unikernels [28] on edge devices with limited resources. The main difference between hypervisor- and container-based virtualization is OS sharing. Lightweight virtualization (i.e., at the OS level) is characterized by its low initialization time, small memory footprint and image size. Thus, unikernels and containers are more agile/portable to be moved than VMs [26].

2.3.1 Network resource virtualization

Traditional network functions based on proprietary middleboxes have several drawbacks, mainly in terms of high capital, operating and maintenance costs, high power consumption, long deployment cycles, and the need for expert staff to manage the plethora of existing types. Over the past decade, network technologies have evolved to offer a set of enabling technologies that leverage the advantages of IoT/Edge/Cloud environments [29], with the promise of supporting energy-efficient IT solutions. This leads to the evolution towards network architectures based on network functions virtualization (**NFV**), which aims to replace middleboxes with software instances running on general-purpose virtualization solutions, such as containers and virtual machines. Thus, NFV replaces the use of traditional network functions to deploy virtual network functions (**NVFs**). This solution also responds to the evolution in networks required by modern technologies such as B5G and IoT, which seek more flexible, scalable and sustainable networks. NFV and software-defined networks [3] allow the management of resources and VNFs within a network slice, independently and individually to fulfil the wide variety of demands requested by different web services, application domains, customers, or operators.

VNFs can run in any computing device (i.e., mobile phone, edge device, cloudlet, etc.), and can be bound together into Service Function Chains (**SFCs**) of connected (virtual) network services.

2.3.2 Service Function Chain

A Service Function Chain is a mechanism that enables the connection of multiple service functions in a sequential manner, forming a service chain. It deviates from the conventional destination-based forwarding commonly used in IP networks. While it shares conceptual similarities with Policy-Based Routing in physical networks, SFC is predominantly regarded as a technology within the domain of *Software-Defined Networking* (**SDN**) [3].

The problem of deploying SFCs/VNFs inside a virtual network slice, satisfying a set of demands and device constraints is known as VNF placement, and it is an important area of recent research [30]. In addition, being conscious that the transition to a fully virtualized network infrastructure will pay a high energy cost

[9], energy efficiency is one of the key innovative targets of future communication systems. SFC placement is a complex undertaking that has been demonstrated to be NP-hard [31].

2.4 Edge Computing

Edge Computing is a distributed computing paradigm that calls for processing the data at the edge of the network, closer to the data sources. The popularity of the IoT [32] and *cyber-physical systems* [33] has led to an ever-increasing number of smart devices connected to the Internet. This results in large-scale data, which compromises traditional cloud computing solutions due to issues such as bandwidth overload, energy consumption, slow response time, poor security and poor privacy. EC has the potential to meet the needs of today's systems in terms of response time, battery life limitation, bandwidth cost savings, as well as data security and privacy. This means that some of the data processing and storage will move to devices closer to where data and services are produced and consumed.

2.4.1 Mobile Edge Computing

Mobile Edge Computing (**MEC**) is a specific application of EC that is closely associated with 5G/B5G (but not exclusive to it). MEC extends cloud computing capabilities and an IT service environment to the network's edge (base stations, central offices, etc.), with the primary objective of enhancing the user experience through reduced latency and alleviated network load. MEC is widely recognized as a pivotal factor in enabling massive IoT deployments and plays a critical role in processing vast datasets originating from an ever-growing number of interconnected devices.

One of the key aspects that sets mobile edge computing apart is its ability to leverage code offloading. Code offloading is a technique that extends the native capabilities of mobile devices by migrating resource-demanding tasks to resource-intensive surrogates. These surrogate devices can be located either at the edge or in the cloud. Code offloading offers several advantages, including enhanced computational performance, flexibility, and storage capacity provided by surrogate processing units. It enables the execution of applications that are not inherently supported by mobile devices and contributes to extending the battery life of such devices by offloading computationally intensive tasks to external systems.

2.4.2 Workload orchestration

Orchestrators are tools that control the behaviour of containers and can automate the deployment, management and scaling of container-based applications [34]. Orchestrators are a fundamental part of environments where many containers, that provide different services and that are deployed on different devices, must be managed. Orchestration tools (e.g., Kubernetes, Docker Swarm, Apache Mesos, etc.) facilitate the automation and scaling of container-based workloads for live production environments. Orchestrators help to optimize resource allocation, load balancing, and fault tolerance, ultimately improving application performance and availability. Additionally, they simplify the management of containerized applications, allowing developers to focus more on application logic and less on infrastructure concerns. Orchestration plays a vital role in enhancing the efficiency of various IT processes, including server provisioning, incident management, cloud orchestration, database management, application orchestration, and numerous other tasks and workflows.

One of the main components of orchestrators is the scheduler, which is responsible for deciding on which device each piece of software will run considering several factors. However, conventional orchestrators typically do not take energy consumption into account in making this decision.

2.4.3 Task scheduling and resource allocation optimization methods

Task scheduling refers to the process of assigning computational tasks to appropriate edge devices or cloud servers in order to optimize system performance. It involves determining which tasks should be executed where and when, taking into account various factors such as resource availability, workload requirements, network conditions, energy consumption, latency requirements, costs, and overall system efficiency [7].

Achieving efficient task scheduling is a complex task due to the distributed nature of the infrastructure and the heterogeneity of tasks and devices. The deployment of pieces of software can be modeled as a resource allocation problem. The goal is to find a deployment solution that satisfies a series of constraints (application's functional requirements, QoS), while minimizing or maximizing an objective function (energy consumption or/and latency). CPSs typically have limited energy resources, and excessive computation can drain their batteries quickly. Thus, task scheduling involves a trade-off between energy consumption and latency. On the other hand, meeting latency requirements is crucial for real-time applications. Balancing these factors requires careful consideration and optimization.

There are several methods commonly used to solve task scheduling optimization

problems (which has been proved to be NP-hard[35]), including solvers, heuristics algorithms, and machine learning techniques. Each of them has its own strengths and limitations, and the choice of the most suitable method depends on factors such as problem size, constraints, execution time, kind of variables involved, and desired optimization objectives. The most used solvers in task scheduling include *Integer Linear Programming*, *Mixed Integer Linear/Nonlinear Programming* (MILP/MINLP) and SMT solvers [30, 7]. While heuristic algorithms and machine learning techniques do not ensure to return a solution (even if it exists) and may provide sub-optimal solutions, ILP, MILP/MINLP and SMT solvers always return the solution or, conversely, report the infeasibility of the problem, and are able to return optimal solutions. ILP and MILP/MINLP solvers differ in the type of variables and the degree of equations supported: in ILP, all variables are restricted to take integer values and equations must be linear; MILP and MINLP can support other types of variables, but at least one variable must be integer-valued, supporting equations that are linear/nonlinear respectively. While SMT solvers do not restrict the type of variables and support nonlinear equations, MILP solvers provide solutions significantly quicker than SMT solvers [36]. Reinforcement learning, deep learning, and deep reinforcement learning are commonly adopted machine learning techniques. However, recent surveys highlight the limitation of machine learning approaches [37, 38] such as uncertainty in finding optimal solutions [38], and the time and energy costs associated with training in supervised learning approaches.

2.4.4 Resource auto-scaling

The management of dynamic service demands is addressed through auto-scaling, which allows for the dynamic allocation of computing resources according to user needs, seeking to maximize performance while reducing costs [39]. This process can be done by increasing/decreasing the number of nodes (horizontal scaling) or by modifying the resources available within the nodes (vertical scaling), the latter not being applied in edge infrastructures. In addition to reducing the energy consumption of the nodes themselves, has been proved that non-computer appliances, such as cooling and lighting, can consume around 33% of the total energy of an infrastructure of up to 500 nodes [40]. Therefore, shutting down nodes where possible can also eliminate considerable non-computing power.

Auto-scaling methods can be classified into reactive and proactive. Reactive solutions use rule-based policies for resource scaling in specific events and employ heuristics for resource allocation. Proactive auto-scaling forecasts future events, like customer request surges, to dynamically adjust resource allocation within a time frame.

There is a prevailing shortage of auto-scaling techniques explicitly designed for edge computing infrastructures. Current auto-scaling solutions developed for the

cloud environment cannot be directly applied to an application operating on an edge cluster without significant adaptations or modifications. The heterogeneity of a limited set of computing resources makes auto-scaling in EC more complex and time-consuming than in cloud systems [41, 42].

2.5 User-centric feature deployment in MEC

In Mobile Edge Computing, a portion of the application is executed on the user's device, which is typically a smartphone. However, users often only use certain features and ignore others. By analyzing user behavior, and considering other parameters like device status (e.g., battery level), we can identify unused or irrelevant features and avoid them to be deployed on the user's smartphone, thus reducing energy consumption.

Self-adaptive software is a type of software system that has the ability to autonomously monitor and adjust its behavior in response to changes in its context, that involves its environment, requirements, or internal conditions. It is designed to adapt and optimize its performance, functionality, or configuration based on the dynamic nature of the system it operates within. Dynamic weaving is a technique often employed in the development of self-adaptive software. It operates by intercepting and modifying the program's bytecode or intermediate representation, enabling the injection of new code, modification of existing code, or application of cross-cutting concerns into the running program.

Specific programming techniques have been used to reconfigure mobile applications, as for example reflection [43], polymorphic methods [44], dynamic proxies [45], Aspect-Oriented Programming [46], and Context-Oriented Programming [47]. Generally, the advantage of employing specific programming techniques lies in their platform independence, thereby enabling their applicability on various mobile operating systems. However, the veracity of this assertion hinges upon the particular technique or technology being utilized.

Chapter 3

Motivation and Approach

In this section, the four Research Questions (**RQs**) that drive this thesis are defined and explained. In summary, we will facilitate the adoption of edge computing in emerging domain systems such as cyber-physical systems and B5G networks, providing a full-stack solution.

3.1 Motivated Research Questions

When it comes to B5G/IoT/Edge/Cloud applications, there is a significant amount of variability to consider. IoT encompasses a wide range of devices, sensors, and platforms, each with its own unique requirements and functionalities. This diversity leads to a multitude of possible combinations and configurations. In addition, the nature of IoT applications requires adaptability to different use cases, environments, and user preferences. Factors such as device capabilities, network connectivity options, data processing requirements, and user interfaces can vary greatly across different IoT scenarios. Furthermore, the variability in IoT applications extends beyond the diversity of devices and platforms. Concerning energy consumption, specific features or functionalities within an IoT application may naturally demand more energy than others, and the amount of energy consumed will greatly rely on the host device. Therefore, it is essential to manage data that enables the estimation of this energy consumption, incorporating additional factors into the equation to account for increased variability.

This high variability can lead to the creation of software non supported by the infrastructure. With numerous manufacturers and vendors offering different devices and technologies, ensuring compatibility becomes challenging. Additionally, the rapid evolution of IoT/B5G technology introduces frequent updates and changes to the infrastructure, making it difficult for software to adapt and remain compatible. The variability in processing capabilities, memory, and connectivity

options among devices further complicates software development. Inconsistencies in data formats, security mechanisms, and network configurations within the heterogeneous infrastructure can also hinder software support. Thus, addressing the variability is crucial to develop software that seamlessly integrates with and is fully supported by the IoT/B5G infrastructure.

Applying an SPL [16] approach would make a lot of sense since it has been successfully applied to different domains to explicitly model variability. Consequently, we define the first RQ:

> **RQ1:** How can we model the huge variability of B5G/Edge/Cloud software and hardware infrastructures focusing on relevant energy consumption-related features? Can we ensure that application and infrastructure dependencies are satisfied and there will be enough resources to attend application task demands?

Once configured the infrastructure and the desired application, the functionalities of these selected features can be implemented in various ways, offering different levels of energy efficiency. For instance, certain implementations may prioritize energy conservation by employing efficient data processing algorithms or utilizing low-power sensors and communication protocols while others might prioritize real-time responsiveness, sacrificing energy efficiency for immediate data transmission and processing. By handling multiple implementations for each application feature, it becomes possible to leverage this advantage. However, it also entails the need to evaluate and identify the most appropriate option for each case, considering the characteristics of the host infrastructure.

After determining the set of tasks (implementation of the application's features) that form the application, developers are confronted with the task allocation process while ensuring a specific level of QoS. The energy consumption of the application tasks largely relies on the device they are executed on [48, 7]. Simultaneously, the execution and communication time are contingent upon the capabilities of the devices involved [7]. The challenge lies in devising a deployment solution that optimizes energy consumption while meeting both functional and non-functional application requirements. Functional requirements encompass hardware dependencies like peripherals or sensor units (e.g., a camera, a thermometer), whereas non-functional requirements, such as QoS, demand certain segments of applications to be completed within specific time limits. Accomplishing the latter is particularly complex, as it entails ensuring one or more groups of tasks conclude their execution within designated timeframes, considering the execution and communication latency of the individual tasks within them.

Many existing approaches overlook the individual configuration requirements and unique characteristics of tasks while generating generalized deployment descriptors [49]. This oversight leads to inaccuracies and inefficiencies in the de-

ployment process. Moreover, certain hardware dependencies within applications and their influence on QoS requirements are disregarded, leading to sub-optimal placement decisions that fail to meet customer expectations.

The placement problem is a generalized assignment problem (**GAP**), that is proven to be NP-hard [35]. The proposed solution should not only deliver a valid assignment solution (or indicate when none is possible) but also do so within a reasonable time frame while using less energy than the energy savings it enables. Thus, we define the second RQ:

> **RQ2:** How can we select the most energy-efficient implementations that align with the infrastructure status and ensure Quality of Service (QoS)? How can we automatize the deployment of software functions in B5G/Edge/Cloud infrastructures?

Although addressing RQ1 and RQ2 could enhance the creation and deployment of energy-aware software in B5G/Edge/Cloud infrastructures, their applicability in different scenarios remains challenging. In fact, most task assignment solutions are proprietary, making it challenging to reuse them in different network environments like B5G.

In B5G networks, orchestrators oversee the behavior of general-purpose (containerized) services and can automate the deployment, management, and scaling of applications based on container technology [34]. As standalone entities, industrial orchestration solutions do not consider energy consumption during the deployment process, nor do they guarantee a specific quality of service for a certain number of users.

With the focus on the network function virtualization application domain, and towards its standardization, adoption, and implementation, the ETSI developed a Reference Architecture Framework, a specification that supports the NFV Management and Orchestration (**MANO**) [50]. The ETSI NFV MANO does not establish specific standards for the implementation of these managers or their coordination interfaces. However, the industry has responded by offering open-source projects that align with the ETSI MANO architectural blueprint [51, 52]. Some notable examples include OSM [53], OPNFV [54], ONAP [55], and Cloudfy [56].

While a correct allocation of tasks reduces energy consumption due to the execution of tasks (dynamic energy consumption), the infrastructure nodes consume energy by remaining on (base energy consumption). In edge infrastructures, reducing the base energy consumption implies increasing/decreasing the number of active nodes. There is a prevailing shortage of auto-scaling techniques explicitly designed for edge computing infrastructures, as current auto-scaling solutions developed for the cloud cannot be directly applied to edge infrastructures [41, 42, 57, 58].

Knowing the alternatives for edge/cloud resource orchestration, we can formulate the following RQ:

> **RQ3:** How can we integrate energy awareness in resource orchestration techniques to guarantee QoS and achieve energy-efficient deployments in next-gen networks?

In Mobile Edge Computing, a portion of the application is executed on the user's device, which is typically a smartphone. However, it is not uncommon for users to only utilize certain functionalities of the application while leaving others unused. By analyzing user patterns and preferences, it can be identified which functionalities are not frequently used or not relevant to the user's needs. By omitting the deployment of these unnecessary functionalities on the user's smartphone, we can significantly reduce energy consumption. In addition to user behavior, the device status (battery level, network status, etc.) plays also a vital role in this adaptation process, forming the device's context. This adaptive approach ensures that the device only runs the essential components of the application, optimizing energy efficiency and prolonging battery life. To minimize energy impact and maintain a high-quality user experience, it is essential for the adaptation process to be both energy efficient and transparent to the user.

In addition, the mobility problem is a significant challenge in MEC scenarios. It involves the task of maintaining uninterrupted connectivity and efficient resource allocation as mobile devices transition between edge or cloud devices. As users move, their devices must establish new connections and adapt to varying network conditions and device capabilities. By having the complete application stored on the user's device, it becomes independent of the availability and stability of the network connection. This means that even if the user temporarily loses network connectivity or switches between different edge or cloud devices, they can still continue to use the application without interruptions.

Several programming techniques have been employed to reconfigure mobile applications, including reflection [43], polymorphic methods [44], dynamic proxies [45], Aspect-Oriented Programming [46], and Context-Oriented Programming [47]. These specific programming techniques offer the advantage of platform independence, allowing them to be applicable to different mobile operating systems. However, the validity of this claim depends on the specific technique or technology being utilized. Consequently, we define the fourth and last RQ:

> **RQ4:** What methods can be utilized to dynamically adapt user applications based on contextual information, while maintaining transparency to the user and optimizing energy efficiency?

3.2 Approach Overview

This chapter also provides a general overview of the incremental approaches used to address RQs [1-4]. Our proposal, depicted in a general manner in Figure 3.1, encompasses distinct processes of Domain, Configuration, Application, and Evolution Engineering, which are explained in the subsequent sections.

Within our approach we define two different roles: (1) *domain experts*, who are responsible for defining the assets of the *Domain Engineering* process, and (2) the software architects (infrastructure managers), who use those assets, reusing and instantiating them for each application and infrastructure in the *Application* and *Configuration Engineering* processes. Note that domain experts do their work only once. On the other hand, infrastructure managers reuse the work done by the domain experts when applying the approach to a specific infrastructure and set of applications. In the *Configuration Engineering* phase, infrastructure managers benefit from the solutions provided in this thesis to easily consider the infrastructure capabilities during the software configuration. *Evolution Engineering* process, both domain experts and application engineers are involved. While domain engineers are responsible for updating the reusable artefacts of the *Domain Engineering* process with new changes, application engineers are in charge of providing the new application requirements to propagate the evolution changes automatically through the artefacts of the Application Engineering process.

3.2.1 Domain Engineering

This step of our research focuses on modelling the variability of IoT/Edge/Cloud/B5G infrastructures and applications using SPLs. Variability modelling in SPL is commonly accomplished through the utilization of feature models (FM), which serve as a fundamental framework for capturing and representing the variabilities inherent within the system.

To deal with the high variability presented in applications and deployment infrastructures, our proposal uses multi-layer FMs [59], along with FMs with cardinality [60] and numerical attributes [61]. In a multi-layer feature model, features are categorized into distinct layers that represent varying levels of abstraction, typically corresponding to different aspects or dimensions of the software system. FM with cardinality allow for more granular relationships between features beyond binary dependencies, enabling features to have specific multiplicity constraints such as multiple selections or required associations with a certain number of related features. Finally, numerical features represent measurable characteristics or parameters, encompassing a wide range of quantitative attributes relevant to the system. By means of *cross-model constraints*, that establish relationships between features that are not explicitly captured within a single FM, our proposal maintains

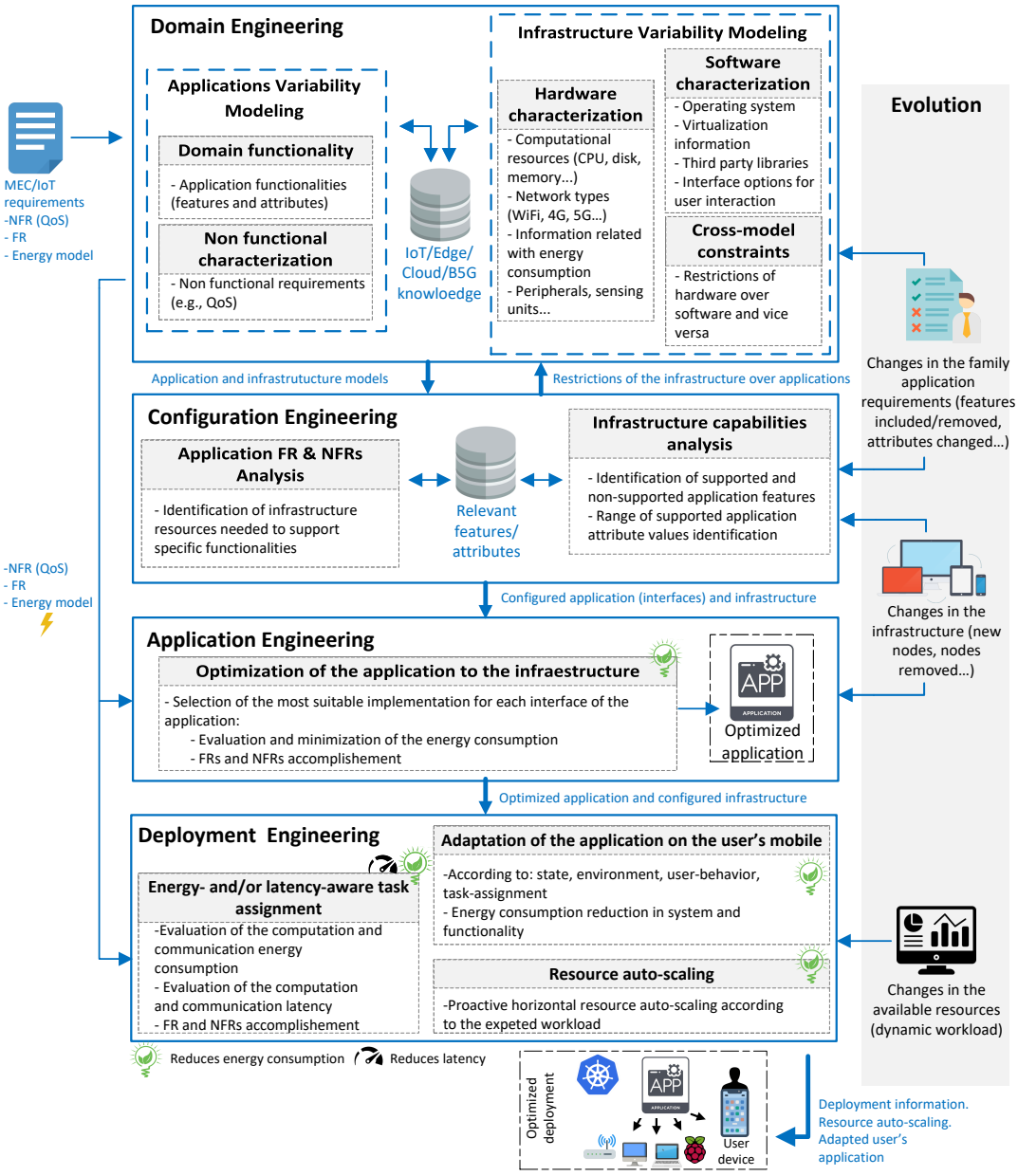

Figure 3.1: Overview of the approach.

consistency between the different models. All this together allows for modelling the variability of the IoT/Edge/Cloud/B5G infrastructure and the application to be deployed.

Our approach uses three FMs in total: one to model the application and two for the infrastructure. While the application FM will depend on the specific appli-

cation, the FMs of the infrastructure can be extended and reused to be applicable to any scenario.

Infrastructures are formed of several heterogeneous devices, described by a set of hardware and software characteristics: type of device, computing power, peripherals, network capabilities, operating system, third-party libraries, etc. These characteristics relate to the type of services/tasks that can be deployed on them, as they refer to the hardware and software requirements demanded by the applications to the host devices.

Regarding the FM of the hardware infrastructure, it contains several features that model the resources of the nodes. These features will determine the latency of execution and communication, as well as the physical resources (e.g., memory) to allocate the tasks. The FM also contains information used to estimate energy consumption, both for execution and communication. For instance, CPU characteristics such as frequency directly influence the energy consumption of computing tasks [62] and factors such as the upload/download transmission powers [7]. Our model also has an attribute which makes it possible to establish the importance of saving energy at the nodes according to the needs of the infrastructure. Therefore, unlike other approaches that have a set of nodes on which the reduction of energy consumption is focused [7], our approach uses a weight-based system to provide flexibility to the deployment solutions and to ease the definition of deployment policies. An example of practical use is to prioritize the execution of tasks on devices powered by solar panels, therefore reducing the whole carbon footprint of the deployment.

On the other hand, the FM of the software infrastructure models software components like the operating system, third-party libraries, and also the interface options for user interaction (e.g., text introduction, voice recognition, etc). In addition, software virtualization (i.e., virtualization and containment technologies supported), which extends software device capabilities, is also considered, distinguishing between hardware-level virtualization and lightweight virtualization based on containers (OS-level) [26]. The same physical device can include several VMs, that are constrained by the hardware resources of the device. Regarding network connectivity, devices may have one or more virtual networks associated.

Hardware and software characteristics are highly dependent, being the software conditioned by the hardware. Such constraints are modelled using *cross-model constraints*, which establish relationships between features that are not explicitly captured within a single feature model. For instance, the ways in which the user can interact with the device (e.g., through a touchscreen) are restricted by the peripherals of the hardware configuration. Despite that, we consider the presence of some hardware characteristics insufficient to assure an interaction method (e.g., acoustic sensing motes do not allow voice recognition). Apart from these types

of constraints, the main way in which software is constrained by hardware is in virtualization, where virtual resources are constrained by physical resources.

3.2.2 Configuration Engineering

Due to the restrictions imposed by the infrastructure on the services that can be provided, it is important to ensure that the created software product can be deployed on the target infrastructure. In turn, when a specific service needs to be offered, is essential to know the devices that the infrastructure administrator must acquire in order to be able to provide that service. Therefore, in the *Configuration Engineering* phase, we present two different solutions to analyse (1) the functional and non-functional requirements (**FR** and **NFR** respectively) accomplished by the infrastructure; and (2) the restrictions imposed in the applications by the infrastructure.

Specifically, the *Application Variability Adapter* module informs the infrastructure manager about the restrictions that a particular infrastructure imposes on the application. The goal is not to add more nodes to the infrastructure, but to constrain the variability of the application (i.e. those features that are optional or adaptable) to the infrastructure capabilities and resources. The aim of this module is to find out whether the application's features are supported by a given infrastructure. Concretely, this module maximizes the number of supported features and upper range of the attributes, while minimizing their lower range, and maximizes the number of supported features. Nevertheless, sometimes it is necessary to provide a specific service, and it is the infrastructure that must be adapted. This is the aim of the *New Devices Finder*. Starting from a specific (configured) application and an infrastructure that can be evolved to meet (unsupported) application features, the goal of this module is to define the characteristics of the minimum set of IoT/Edge/Cloud devices that are not included in the current infrastructure, yet needed to support the application.

Both modules are posed as constraint satisfaction problems. Concretely, we rely on Z3, an SMT (Satisfiability modulo theories) solver [63], to obtain the solutions. Thus, (1) the algorithm always returns a solution (unlike heuristic algorithms), which guarantees that the deployment is feasible or the impossibility to deploy the application if no solution is found; and (2) a large number of constraints (required to solve the problems at hand) help SMT-solvers to reduce the search space and to find the optimal solution faster [64, 65]. Unlike solvers that use ILP/MILP/INLP/MINLP, SMT solvers neither restrict the types of the variables involved in the problem formulation (that are unknown, as our modules' inputs are derived from the FMs) nor the degree of the equations. Additionally, it is proved that Z3, and concretely its optimization module (νZ), returns optimal solutions [65]. Nevertheless, the flexibility of our approach may allow the use

of any other mathematical model (that does not restrict the variable types) for optimization.

3.2.3 Application Engineering

The *Application Engineering* process (third step of Figure 3.1) aims to generate software products based on the specific application's requirements. This phase, together with the *Domain* and *Configuration Engineering* phases, address RQ1.

3.2.3.1 Task Mapping

Once configured, the application's FM obtained from the *Configuration Engineering* phase is mapped with the appropriate tasks' interfaces of the application. Each feature of the application's FM is related to one or more task application interfaces. The interfaces that form the application are modelled as a *task-call graph*, which is typically a finite directed graph with no directed cycles [7]. The set of vertices represents the application tasks and the edges represent their dependencies. This representation allows us to consider tasks whose beginning of execution depends on a previous task (sequential dependency) and the existence of parallel tasks and also group the interfaces in sets with sequential dependency and maximum time to be completed, thus ensuring the QoS of the application in the deployment phase.

3.2.3.2 Application-to-infrastructure optimization

The same application functionality (interface) can have multiple implementations, and its efficiency, in terms of latency and energy consumption, greatly depends on optimization for the characteristics of the execution node. For instance, leveraging the native support of arithmetic operations by the arithmetic logic unit of processors can lead to improved efficiency.

Hence, our proposal involves handling multiple implementations for each application functionality. Upon configuring the FM of the application, the *Task Implementation Selector* module evaluates each task implementation, ensuring that the application requirements are met and estimating the overall energy consumption of the task set. This process aids in identifying the task implementation that demands the least energy for each application functionality, considering the features of the infrastructure's nodes. Additionally, the module verifies the deployment feasibility for a given number of users, taking into account the initial status of the nodes configured using the aforementioned FMs. As the node status may change over time due to factors such as available devices and workload variations, continuous monitoring of the infrastructure is crucial. Consequently, the

Tasks Implementation Selector module selects the most suitable implementation for each application feature based on the characteristics of the infrastructure.

3.2.4 Deployment Engineering

The *Application Engineering* process (fourth and last step of Figure 3.1) aims to efficiently deploy the application (RQ2), adapt the user's application according to the task assignment (RQ3) and orchestrate edge resources (RQ4).

3.2.4.1 Task scheduling optimization

In this step, the application's tasks, already optimized to the infrastructure in the *Application Engineering* process, are assigned to the nodes of the infrastructure. The objective is to optimize the deployment in order to minimize dynamic energy consumption and/or latency while ensuring the functional and non-functional requirements of the application.

To this end, the output of the *Application Engineering* process is used to select the infrastructure device that best suits the execution of each task in order to minimize dynamic energy consumption. For this purpose, the result of the previous phase, together with information on the infrastructure and its current status obtained from the *Configuration Engineering* step, is used by a deployment optimization module. During the thesis, several solutions of the deployment optimization module have been developed [66, 13, 12], the most advanced being the ones presented in [13], [12] and [14]. While [13] and [12] rely on an SMT solver (Z3) to obtain the solution, our last release [14] uses the *Gurobi Parallel Mixed Integer Programming* solver. The facts for this choice are: (1) equations included in the task assignment algorithm are linear, so using nonlinear solvers would add meaningless complexity; (2) supports the types of variables that need to be managed, which are now known, in contrast to the Configuration Engineering phase where they were uncertain (3) unlike heuristic algorithms, MILP solvers always return a solution, which guarantees that the deployment is feasible or the impossibility to deploy the application if no solution is found; (4) MILP solvers provide solutions significantly quicker than SMT solvers [36]; and (5) Gurobi is especially strong in solving MILP problems [67], supporting multi-threading and aiding problem scalability. Nevertheless, it could be possible to use other solutions to our approach.

Concretely, [13] presents the *Optimal Task Assignment Framework*, which selects the nodes to which each task should be assigned in order to minimize dynamic energy consumption. This module takes into account both the energy consumption of execution and communication, using widely adopted energy models to predict energy consumption [7, 48]. In [12], we present the *Energy Consumption*

and Latency Minimizer (ECLAM), which extends the functionality of the previous module allowing to minimize the latency of the applications. Specifically, this module returns allocation solutions that minimize energy consumption, latency, or both by returning a trade-off between energy consumption and latency [63]. In [14], we present the *Energy Efficient Task and Resource Allocator* (iTAREA), which allocates, apart from RAM, disk, peripherals and bandwidth, the portions of CPU to each task in order to accomplish a certain QoS.

All modules solve a GAP, which is proved to be NP-hard [35] being therefore computationally expensive. In addition, the time needed by our modules to provide a solution varies according to the size of the problem [68]. Thus, we also evaluate the applicability of our modules for different infrastructure and application sizes using a Benchmark. For all modules, we conclude that they take a reasonable amount of time to bring a solution, being feasible to use in real environments.

3.2.4.2 Integration with containerization and orchestration technologies

Our proposal has been integrated with deployment technologies widely used in the industry, such as Docker (containerization) and Kubernetes (orchestration). This integration provides deployments with the high availability and fault tolerance that these tools supply. For this purpose, the deployment and resource reservation solution obtained in the Application Engineering process is s included in files that can be understood by these technologies, which allows for limiting the use of resources by themselves. In this thesis, we have demonstrated that such integration is fruitful and that the solutions presented throughout the thesis can reduce energy consumption while ensuring a certain QoS in real scenarios.

In [14] our approach is integrated as an extension of the Open Source Management and Orchestration (OSM) project [53], an ETSI-sponsored project to develop an open source NFV MANO software stack aligned with ETSI NFV and tested in real scenarios. OSM provides a VNF manager, an NFV orchestrator and supports connection to third-party Virtualized Infrastructure Managers (VIMs) (e.g., OpenStack, OpenVIM, VMWare). In addition, OSM supports container technology (via Kubernetes [69]) to enable the design and implementation of Kubernetes-based Network Functions (KNFs) [53] running inside containers. Our solution, named HADES, extends OSM by enabling the placement and configuration of VNFs/KNFs and their subsequent resource allocation and deployment at the edge, minimizing energy consumption and latency. Furthermore, HADES enables deployments that take into account multiple VIMs, a feature not natively supported by OSM. Furthermore, the use of Kubernetes endows the infrastructure with fault tolerance and high availability.

The HADES architecture, depicted in Figure 3.2, follows the N-MAPE-K

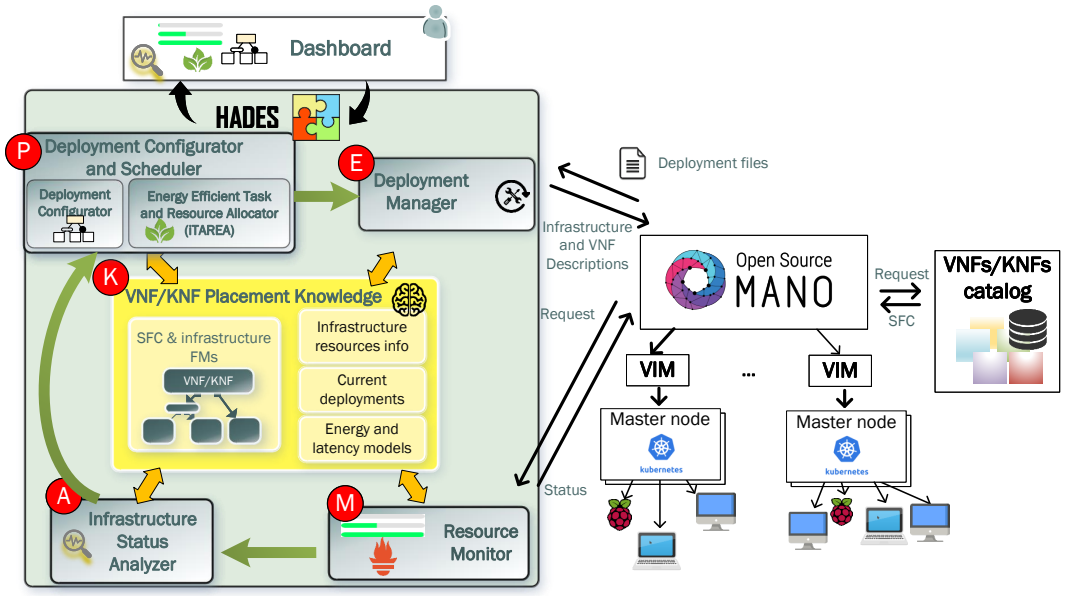

Figure 3.2: General overview of HADES

loop, an extension of the classical Monitor-Analyze-Plan-Execute loop over a shared Knowledge (MAPE-K, [70]) adapted for Networks, defined in the DAEMON project [71]. MAPE-K is one of the most influential reference control models for autonomous and self-adaptive systems. We will describe the HADES components implementing the N-MAPE-K loop:

- ***M****onitor*: The *Resource Monitor* module (M in Figure 3.2) monitors the resource usage of the nodes (CPU, memory, disk, and links' bandwidth) in real-time using Prometheus and Kubernetes monitoring tools. This is the first step of the N-MAPE-K loop, the one in charge of collecting the information that feeds the loop. This information is persistently stored as a time series in the *K* component, which is the central part of the loop storing all the information needed by each N-MAPE-K component.

- ***A****nalyze*: This is the second step of the loop. The output of *M*, that is, the current status of the infrastructure is analyzed (in component *Infrastructure Status Analyzer* labelled as A) to check the availability of the infrastructure's capabilities and check if there is a more energy-efficient allocation of existing deployments (taking into account the cost of migration).

- ***P****lan*: Once, the A function decides a re-deployment of the VNFs should be performed, the *P* task is invoked to generate a new optimal solution adapted

to the current network status. The *Deployment Configurator and Scheduler* (labeled as P) contains the *energy effIcient TAsk and REsource Allocator* (iTAREA), which is responsible for computing the optimal (energy-aware) placement of VNF/KNF by executing the iTAREA algorithm. The information about the amount of resources (CPU, RAM, disk, and bandwidth), the resource requirements of applications, and current characteristics of the infrastructure (e.g., the list of nodes powered by renewal energy) are taken from the central knowledge K module, keeping this information persistently in a relational database. We use a formal variability model called Feature Model (FM [72]) to specify IoT infrastructures and generate valid configurations (see the Deployment Configurator component), and also to specify application features and their demands [13]. Also, the SFC is modelled with a graph that specifies the sequence of VNFs and data flow.

- ***E**xecution*: Finally, this is the last step of the loop. The VNF/KNF *Deployment Manager* module (component E) executes the deployment (placement and resource allocation) computed in the previous step. To do so, it uses the OSM's North Bound Interface API to send the deployment file containing the KNF/VNF new locations to OSM. OSM's North Bound Interface is RESTful and follows the ETSI SOL005 standard.

- ***K**nowledge*: The central *VNF/KNF Placement Knowledge* component (tagged with a K) maintains the information retrieved from KNF/VNF repository, the KNF/VNF configuration models, the energy and latency models, monitored infrastructure's status, and software and hardware infrastructure characteristics (CPU power, peripherals, memory, etc.).

In addition, HADES includes a Dashboard web-based component, which provides a graphical interface so that the user can control, monitor and consult the data and activities of the components of the N-MAPE-K loop (see Figure 3.2).

3.2.4.3 Resource auto-scaling

Managing the dynamic needs of users' service requests is addressed by an auto-scaling mechanism that seeks to maximize performance by allocating computing resources adapted to current demands. This is a challenging problem, especially for Edge Computing environments due to their heterogeneity and dynamic nature. In fact, proactive auto-scaling solutions for edge computing systems are scarce. Due to the heterogeneity of computing resources, proactive auto-scaling for edge computing is more complex and time-consuming than in cloud systems [41, 42]. So far, the feasibility of cloud-native auto-scaling techniques in edge computing

has not been evaluated and should take into account the characteristics of IoT devices, not just virtual machine instances and containers [41, 42].

Despite these challenges, this thesis takes a first approach to the problem/issues of horizontal auto-scaling and energy efficiency. It proposes a proactive horizontal auto-scaling framework for edge infrastructures, which takes into account both the baseline (idle) and dynamic (due to application execution) energy consumption of edge nodes, as well as the node scaling mechanism. The proposed auto-scaling solution has four resource reservation policies, applicable to two different operating modes (i.e., a total of eight configurations), each of which can be used with the desired tasking policy.

The auto-scaling solution is applied to an edge-based infrastructure that serves the real dynamic edge workload provided by Shanghai Telecom [73]. To do so, the auto-scaling module is connected to the open-source EdgeCloudSim edge environment simulator [74], applying the proposed auto-scaling approach to three different workload scenarios: ascending, descending and fluctuating.

The results show up to a 92.5% decrease in energy consumption, a failed request rate of up to 0%, and reasonable execution times of the auto-scaling process for different problem sizes.

3.2.4.4 Application adaptation on the user's device

In MEC, the last link in the chain where energy consumption can be minimized is in the application that runs on the user's device. A portion of the application is executed on the user's smartphone. However, analyzing user patterns and preferences allows the identification of unused or irrelevant functionalities. By excluding these unnecessary components from deployment, we can significantly reduce energy consumption. Adapting applications according to user behaviour adjusts the number of tasks performed by the applications to the ones that are relevant to the user at a certain moment. In addition, such mechanisms are a solution to the mobility challenges for the realization of ubiquitous and reliable (i.e., seamless and error-free) computing [7], since it is possible for the user to leave the radius of action of the edge infrastructure. As a solution, we propose to have the entire application on the user's mobile and adapt the functionality of the application according to the user behaviour. For a seamless user experience, it is crucial for this energy-efficient adaptation process to be transparent to the user.

We propose a generic adaptation schema inspired by the MAPE-K loop [75], followed by four different adaptation engines (**AE**) for Android applications. Specifically, such adaptation mechanisms are **AE1: Internal Proxy**, **AE2: External Proxy**, **AE3: Xposed Proxy** and **AE4: Xposed**, differing mainly in how and when the application is adapted. When adaptable classes are implemented as dynamic proxies (AE1 and AE2), the proxy works as a class handler, controlling the

access to the class functionality. This time, the adaptation is made at runtime. Using the Xposed framework (i.e., virtual method hooking technique [76]), AE3 and AE4), the application is modified just before it starts.

Through our results, we have demonstrated that the adaptation engines can reduce the energy consumption adapting the application to the user behaviour by up to 20%. In addition, the use of the adaptive mechanisms increases the energy consumption of the application by 2.51% in the worst case, making it energy-efficient, being a good solution to be applied in MEC environments.

Chapter 4

Discussion of Results

This chapter answers the research question by discussing this thesis's main results and contributions.

Our **first contribution** starts with **the definition of two extensible and reusable variability models for B5G/Edge/Cloud infrastructures, specifically designed to estimate energy consumption.** To deal with the high variability presented in applications and deployment infrastructures, we use multi-layer FMs [59]. We propose the use of FMs with cardinality [60] and numerical attributes [61]. By means of *cross-model constraints*, we maintain the coherence between the FMs of different layers. We use three FMs: one to model the application and two for the infrastructure (splitting hardware and software characteristics). While the application FM will depend on the specific application, the FMs of the infrastructure can be extended and reused to be applicable to any scenario. These models contain relevant information for the calculation of the energy consumption of the computation and communication tasks. ***This contribution****, which has been extended and improved over the course of the thesis, can be found in [77, 13, 66], although its most advanced version* ***is presented in our highlighted work [12] (Chapter 7)****.*

Our **second contribution** are **two software modules to aid developers during the configuration and subsequent deployment in edge-based infrastructures of distributed applications using SPLs**. These modules analyze and consider the dependencies between applications and infrastructure to ensure that sufficient resources are available to meet the demands of the application tasks. Concretely, the *Application Variability Adapter* (AVA) module adapts the variability of the application according to the capabilities of the infrastructure. this module informs about the new devices needed to run a specific application. The *New Devices Finder* (NDF) module, whose objective is, for a specific infrastructure, to inform about the new devices needed to run a specific application. To implement the AVA and NDF modules we opted for an SMT solver (specifically

Z3 in its version for Python) mainly because of its ability to handle variables of all types (necessary because our modules use as input the data from the FMs of the application and the infrastructure, whose type of variables are uncertain). The applicability of the four modules is evaluated by testing their execution time for different problem sizes using a Benchmark version of each module. *This contribution and these results are elaborated in our highlighted work [12]* (Chapter 7).

A1: Using multi-layer feature models we can model the extensive variability of B5G/Edge/Cloud software and hardware infrastructures, with a specific focus on features related to energy consumption. Through our AVA and NDF modules, we address application and infrastructure dependencies, ensuring that sufficient resources are available to meet the demands of application tasks. Their applicability to real-world scenarios and large problem sizes has been extensively validated.

Our **third contribution** starts with the definition of **a software module that determines, from a set of implementations of an application functionality, the most energy-efficient**. To provide this fine-grained adaptation of the application to the infrastructure, we define the *Tasks Implementation Selector* (TIS) module, which uses the resulting task-call graph of mapping FM's features to the application's functionalities, along with the deployment infrastructure configuration to highlight the most appropriate implementation (among those that the system handles) for each functionality of the application. Then, an optimal task assignment framework assigns each application task to the most suitable edge node for its execution minimizing the energy consumption and assuring the application QoS.

Our **fourth contribution** is a **task-assignment solution that uses the information collected by the application and infrastructure FMs and the output of the TIS module, applicable to any application and infrastructure**. Using energy and latency models that calculate the energy consumption/execution time according to the characteristics of the nodes and tasks, the deployment descriptor transcends infrastructure and application dependencies. In the literature, a wide range of techniques is utilized for solving the task-assignment problem, including ILP variations, SMT solvers, heuristics, and machine learning methods (see Section 2.1.3). During this thesis, several versions of the optimal task assignment module have been presented, making use of different energy and latency models. In the first release of the module, after thoroughly assessing various solvers we chose to utilize an SMT solver, specifically Z3 in its Python version. The decision was based on its capability to handle variables of all types and in its ability to allow divisions between variables, necessary to apply the energy model. However, when the variable types are restricted to boolean and numeric variables,

MILP solvers serve as a faster alternative [36], and was applied to other releases of our module. The experimental results demonstrate significant reductions in energy consumption, with approximately 41%-62% reduction in user node consumption and 34%-48% reduction in devices involved in task assignment. Additionally, our modules, which have been tested in real-world scenarios, exhibit efficient execution times across different problem sizes, as evaluated using a benchmark, validating the applicability of our proposal. ***This contribution and these results was premiered in our highlighted work [13]*** (Chapter 8).

A2: The TIS module allows fine-grained optimisations of task implementation to the infrastructure to reduce energy consumption. The resulting task-call graph of the TIS module, together with the description of tasks and infrastructure represented in the variability models, allow to apply application- and infrastructure-independent latency and energy models and automatize the deployment of software functions in B5G/Edge/Cloud infrastructures.

Our **fifth contribution** **is a horizontal auto-scaling solution for edge infrastructures with different operation modes to reduce energy consumption, applicable to different contexts**, and adaptable to the needs of the infrastructure/service. It is formed by two main components: a workload predictor, responsible for forecasting the future workload based on historical data through machine learning; and the *Essential Node Identifier* module, responsible for deciding the nodes that must be kept active to satisfy the expected workload and minimize the energy consumption (that relies on the SMT solver Z3). Its effectiveness and applicability are validated by applying it to a simulated edge infrastructure that serves the real edge workload provided by Shanghai Telecom in three different scenarios: increasing, decreasing and fluctuating workload. Using the goal-question-metric approach, the results are used to evaluate the impact on energy consumption, how energy awareness affects auto-scaling effectiveness (in terms of the number of failed requests), and how the energy-efficient proactive scaling module behaves to different problem sizes and operation modes. *This contribution and these results are elaborated in our work [78].*

Our **sixth contribution** is **HADES, an extension of the ETSI-hosted project OSM that extends its capabilities by minimizing energy consumption and ensuring the QoS of supporting services.** Additionally, it supports VNF/KNF deployment through OSM by enabling zero and one-touch management operations. The HADES architecture follows the N-MAPE-K loop, an extension of the classical Monitor-Analyze-Plan-Execute loop over a shared Knowledge (MAPE-K, [70]) adapted for Networks, defined in the DAEMON project [71]. After an extensive evaluation of VNF orchestration solutions, it has been de-

termined that OSM stands out as the most advantageous choice for minimizing energy consumption in IoT/Edge/Cloud/B5G infrastructures. This conclusion is primarily based on OSM's low hardware requirements, which contribute to reduced energy consumption and enhanced applicability. Our energy-aware VNF placement solution manages heterogeneous edge infrastructures and considers the energy consumption and the computation and communication delay within an NFV infrastructure. HADES also considers requirements that are not commonly addressed in the VNF placement, such as memory, storage, or a specific hardware/server configuration (e.g., a GPU, a specific network card, etc.). In addition, HADES determines the CPU portions (millicores) that each node should allocate to each task according to the characteristics of the task and the node to ensure a specific QoS. Using a latency model, HADES foresees the resources needed in each node to assure a specific QoS. This significantly increases the problem's complexity. Thus, HADES relies on Gurobi, an MILP solver to return a solution, as the variable types to be handled are restricted to be numeric or boolean. In addition, the energy model used avoids the division of variables, a function that is not allowed in MILP solvers. Through an extensive evaluation, it has been determined that HADES achieves a significant reduction in energy consumption, reaching up to 51% when compared to the default deployment suggested by OSM. Furthermore, measurements of QoS after deployment confirm that HADES successfully meets the desired QoS requirements. Additional experiments conducted in simulated environments demonstrate even greater reductions in energy consumption, with up to 59% improvement compared to five other allocation policies. The proposed solution exhibits minimal energy overhead while delivering substantial energy savings, and it has demonstrated sufficient speed to be applicable in real-world environments, even on standard commercial computers. ***This contribution and these results are elaborated in our highlighted work [14]*** (Chapter 9).

A3: Energy efficient resource orchestration can be integrated into resource orchestration techniques by extending platforms such as OSM to be energy awareness, lowering the energy consumption of deployments and infrastructure. Using a latency model, our deployment descriptor can foresee the resources needed in each node of the B5G/Edge/Cloud infrastructure to assure a specific QoS.

Our <u>seventh contribution</u> is **a generic adaptation schema inspired by the MAPE-K loop along with four distinct adaptation engines (AE) designed for Android applications**. The adaptation engines, namely AE1: Internal Proxy, AE2: External Proxy, AE3: Xposed Proxy, and AE4: Xposed, offer varying approaches for application adaptation in terms of how and when it occurs. In the case of adaptable classes implemented as dynamic proxies (AE1 and

AE2), the proxy acts as a class handler, governing access to class functionality. This adaptation takes place dynamically during runtime. On the other hand, using the Xposed framework, specifically the virtual method hooking technique, AE3 and AE4 modify the application just before it begins execution. However, all these techniques allow to do so transparently to the user. Through our results, we have demonstrated that the adaptation engines can reduce the energy consumption adapting the application to the user behaviour by up to 20%. In addition, the use of the adaptive mechanisms increases the energy consumption of the application by 2.51% in the worst case, making it energy-efficient and a good solution to be applied in MEC environments. ***This contribution and these results are elaborated in our highlighted work [15]*** (Chapter 10).

A4: By the use of an adaptation schema inspired in the MAPE-K loop it is possible to dynamically adapt applications based on contextual information. We have constructed four different adaptation engines that work transparently for the user and have a negligible energy overhead.

Chapter 5

Related Work

This chapter overviews the current state-of-the-art of the different techniques involved in the development of our proposal. We study SPL proposals that take into account the infrastructure during software development. We also compare different techniques and scenarios for workload allocation in heterogeneous infrastructures. We present, and resource orchestration. Finally, we study the self-adaptative software approaches.

5.1 Variability management: SPL and heterogeneous edge infrastructures

In order to use SPL to assist developer towards and efficient deployment in the edge, it is necessary to be aware of infrastructure in terms of variability management and modelling. However, despite its significance, infrastructure issues are frequently overlooked in SPL models including in those approaches where infrastructure plays a central role such as IoT systems [79].

Where incorporated, the modelling of infrastructure-specific features tends to intertwine with other application-specific features, resulting in a direct mapping of application features to individual software components at a 1:1 ratio. The separation of application and infrastructure features into separate models is intended to reflect their possible evolution independently, as well as the constraints that the infrastructure imposes on the deployment of software applications. Although the separate modelling of infrastructure features favours their independence and reuse in different application domains, existing approaches usually allow the configuration of a single device, which precludes their application to infrastructures formed by multiple nodes. Consequently, they are not suitable for application to edge computing environments. This neglect regarding the modelling of platform dependencies also limits the optimization of the software infrastructure, since many

characteristics (e.g., performance, energy consumption) depend on the devices on which the applications are running [7].

Maintaining a single large FM for an entire system is neither feasible nor desirable. Several works [80, 81, 82, 83, 84] propose to define multiple FMs to deal with the variability management of large and complex FMs related to numerous complex constraints. Each FM groups features that represent different viewpoints, subsystems, software system concerns [84], or levels of abstraction, such as platform descriptions [21, 85], or development stages [86].

Although the separate modelling of infrastructure characteristics favours its independence and reusability in different application domains, most approaches only allow the configuration of a single device, which precludes their application to infrastructures composed of multiple nodes [21, 85]. Consequently, they are not suitable for application to real Edge Computing environments. This neglect regarding the modelling of platform dependencies also limits the optimization of the software infrastructure, since many characteristics (e.g., performance, energy consumption) depend on the devices on which the applications are running [87, 88]. Furthermore, the mentioned approaches fail to assist developers in ascertaining the alignment between a configured infrastructure and the intended application functionalities, as well as how to reconcile any disparities between application demands and infrastructure support. The modeling of non-functional requirements inhibits the reusability of the infrastructure's FM across diverse applications and overlooks the coexistence of multiple applications running concurrently on a shared infrastructure, thereby complicating the maintenance and evolution of both the application and infrastructure FMs. Moreover, these approaches primarily concentrate on hardware infrastructure features, neglecting the increasing significance of software variability in CPSs, including virtualization options and the prevalence of IoT middleware in modern contexts. Additionally, they do not offer insights to infrastructure managers regarding the compatibility of a configured infrastructure with the application's functionalities or provide guidance on addressing any mismatches between the application's requirements and the infrastructure's capabilities.

5.2 Workload allocation in heterogeneous edge infrastructures

Proper resource and workload allocation is crucial, especially in edge infrastructures that often contain resource-constrained nodes. In this section we delves into the allocation of general workloads (tasks) especially focused on Mobile Edge Computing (MEC).

In Edge Computing, workloads encompass the tasks or computations assigned

Table 5.1: Comparison of task offloading approaches for heterogeneous devices

Approach	Design objective(s)	Optimization focus	Techniques[a]	Task model	Granularity level	Devices' characteristics considered	Solution mechanism
[8]	Overall energy and latency	User, data flow	TA	Binary offloading	Task	Hardware	Heuristic algorithm
[48]	Energy and latency	User	TA, DVFS	Partial offloading	Service	Hardware	Relaxation-based algorithm
[89]	Successful offloading probability	User	TA	Partial offloading	Task	Hardware	Heuristic algorithm
[90]	Energy	User + Edge	TA	Partial offloading	Task	Hardware	Distributed algorithm admitting the Nash equilibrium
[91]	Latency	User	TA, CC	Binary offloading	Task	Hardware	Distributed matching algorithm
[92]	Cost	User + Edge	TA, CM	Binary offloading	Application	Hardware	Markov decision problem
[93]	Energy and latency	User	TA, CM	Binary offloading	Task	Hardware	Heuristic two-stage algorithm
Our approach [13]	**Energy**	**Customized**	**TA, AA**	**Partial offloading**	**Task**	**Hardware and software**	**SMT based algorithms**

[a] TA: Tasks Allocation; DVFS: Dynamic Voltage and Frequency Scaling; CC: Computation Caching; CM: Computation Migration; AA: App Adaptation

to edge servers for processing. These workloads can vary widely and may include data processing, machine learning, and other computationally intensive applications. In MEC environments, task offloading involves transferring computationally intensive tasks from a user's device to an edge server, aiming to enhance the user experience by reducing latency and alleviating the network load on the user's device, potentially reducing energy consumption.

Approaches found in the literature differ in the number of users supported and in the type of edge nodes considered for offloading. Many approaches consider only one edge node (or a group of them with homogeneous characteristics), focusing on systems consisting of one mobile user (single user) or multiple mobile nodes (multi-user approaches) [7, 6, 8]. Considering only homogeneous nodes, significantly reduces the complexity of the problem, since only the decision to offload code or certain tasks (i.e., execute a specific computational load locally or remotely) is considered. A more realistic but complex scenario involves a heterogeneous set of edge nodes (in hardware and with different types of available resources), shared among multiple users. In this context, alternative approaches at least take task allocation into account. Task allocation encompasses making decisions not only about executing a computational load locally or remotely (binary offloading decisions) but also determining the most suitable node for performing the offloading. As a result, task allocation involves resource management, including the modelling of node resources and the execution of offloading to offer a comprehensive solution.

Table 5.1 overviews a comparison of approaches for task offloading. Concretely, Table 5.1 compares these approaches in terms of: the objective or objectives that drive the task allocation (second column); the type of node(s) for which the optimization is intended (user, user and edge, or customized by the developer; third column); the edge computing techniques applied (fourth column); the task model used (fifth column), which is mainly characterized by a binary or partial offloading scheme [7]. While the binary offloading does not allow the partition of tasks, a partial offloading task model allows splitting tasks into simpler components, providing much more flexible solutions; the sixth column indicates the granularity

level supported by each approach, i.e., the size of the computational load that is considered for offloading (tasks, services, or applications); the seventh column refers to the nodes characteristics that are considered by the tasks allocation algorithm; and, finally, the eighth column shows the method used to find a solution for the task allocation problem.

As Table 5.1 shows, none of the existing works (except ours [13], characterized in the last row of Table 5.1) considers the software characteristics of the infrastructure (such as operating system, software virtualization support or third-party libraries) when deciding task allocation. In addition, our approach enables finer-grained computation offloading than binary offloading by supporting parallel execution between the user's mobile and edge nodes. When possible, our approach is able to adapt application tasks according to the existing edge infrastructure, with heterogeneous nodes, and with the goal of minimizing energy consumption according to the needs of the infrastructure manager or the infrastructure itself.

5.3 Resources orchestration and VNF placement

The varied, ever-changing, and resource-limited characteristics of edge infrastructures demand a versatile orchestration framework capable of dynamically accommodating application QoS requirements. In this section we study the techniques used to manage infrastructure resources in an energy-efficient manner, and the unique scenario of orchestrate Virtual Network Functions (VNFs), with a specific focus on works addressing the service function chain (SFC) placement.

5.3.1 Auto-scaling

The scarcity of auto-scaling techniques tailored specifically for edge computing infrastructures is apparent. A plethora of approaches in the literature address auto-scaling in cloud-based infrastructures with the goal of minimizing energy consumption (or its associated cost) [42, 94, 57]. However, it is reported by several authors that the feasibility of applying cloud-native auto-scaling techniques in edge computing environments has not been evaluated, and cannot be compared by not taking into account the heterogeneous nature present in edge environments [41, 42, 57, 58]. Existing auto-scaling solutions designed for cloud environments are not directly transferable to applications running on edge clusters without substantial adjustments or modifications.

Current auto-scaling methods primarily adopt a reactive approach, utilizing rule-based policies for resource provisioning during specific events. However, this method's sluggishness may hinder the attainment of service response targets. For

workloads that demand frequent scaling, this reactive approach can lead to high costs in terms of execution time, resource utilization, and energy consumption.

In contrast, predictive methods prove more effective in minimizing response time and meeting service latency objectives. By forecasting future workloads, these methods can reduce the number of auto-scaling interventions. Proactive auto-scaling, like the solution proposed in this thesis, remains infrequent in edge computing environments and typically concentrates on horizontal scaling based on predictions of user requests and resource utilization by deployed services [42].

5.3.2 MANO platforms

Since NFV was considered a key technology for the evolution of mobile networks [7], both industry and academia have devoted many efforts to their standardization, adoption and realization. With this aim, the European Telecommunication Standard Organization (ETSI) developed a Reference Architecture Framework, a specification that supports the NFV Management and Orchestration (MANO) [50].

Based on the ETSI-NFV reference architecture, the main components comprise the NFV orchestrator (NFVO), virtual network function manager (VNFM), and virtualized infrastructure manager (VIM). These components handle various tasks, including NFVI management, resource allocation, function virtualization, and placement. Additionally, the ETSI NFV architecture addresses the management of SFCs. The NFVO is responsible for managing the life cycle of SFCs and their VNFs, coordinating actions such as instantiation, update, scaling, migration, and termination in conjunction with VNFMs.

The ETSI NFV MANO does not establish implementation standards for these managers or their coordination interfaces. Nevertheless, the industry has developed some open source projects that follow the architectural blueprint of the ETSI MANO [51][52] such as OSM [53], OPNFV [54], ONAP [55], and Cloudfy [56].

Given the similarity of existing NFV platforms in terms of components and services, we conducted a comparison based on the resources required for their recommended installations. Optimizing the utilization of infrastructure resources is crucial for our objective, especially since edge devices are often resource-constrained, and the amount of device resources used directly impacts energy consumption [7]. An in-depth comparison of the installation and operation requirements shows that OPNFV [54] and ONAP [55] require more vCPUs, RAM and disk than OSM, which has a minimum requirement of 2 vCPUs, 6 GB of RAM and 40 GB of disk [53]. OSM emerges as the most lightweight option, aligning with the objective of minimizing the energy consumption. The significance of low resource usage extends beyond saving energy, considering that edge infrastructures often consist of nodes with limited computational capacity, necessitating lightweight solutions.

Table 5.2: Related works on SFC Placement

Work	Objective(s)[a]	Target infrastructure	Resources considered[b]	Solution mechanism(s)[c]	Flexible resource allocation	VNF reallocation	VNF configuration	Kind of solution
[98]	RO	Edge	CPU, M, B	ILP, HA	×	×	×	Proprietary
[99]	P	Cloud	CPU, M	ILP, HA	×	✓	×	Proprietary
[100]	L, EC	Edge, Cloud	Not specified	IA	×	×	×	Proprietary
[101]	VNRC, BR	Edge, Cloud	CPU, GPU, M, D	IQCP, VLMCE, HNN	×	×	×	Proprietary
[102]	RO	Cloud	CPU, M, B	TPO	×	×	×	Proprietary
[103]	Pe	Edge, Cloud	CPU, M, B	ML	×	×	×	OSM extension
[104]	Pe	Edge	CPU, M, D, B	ILP, HA	×	×	✓	NFV Mano
[105]	Pe	Edge	CPU, GPU, M, D	ML	×	×	×	Proprietary
[106]	RO	Edge	CPU, B	MILNP, HA	×	×	×	Proprietary
[107]	EC	Cloud	CPU, T, B	DTS	×	×	×	Proprietary
[108]	EC	Cloud	CPU	HA	×	×	×	Proprietary
[109]	EC	Edge, Cloud	CPU, M, B	HA	×	×	×	Proprietary
[110]	EC	Edge	CPU (cores), B	NLIP, MVA	×	×	×	Proprietary
[111]	EC	Cloud	B, AbR	ILP, HA	×	×	×	Proprietary
Our approach	**EC**	**Edge, Cloud**	**CPU, GPU, M, D, B, T, Per**	**MILP**	✓	✓	✓	**OSM extension**

[a] BR: Blocking ratio; EC: Energy consumption; L: Latency; Pe: Performance; RO: Resource optimization; VNRC: VNR cost.
[b] AbR: Absolute resources; B: Bandwidth; D: Disk; M: Memory; Per: Peripherals; T: Node type.
[c] DTS: Decision-tree search; IA: Iterative algorithm; HA: Heuristic Algorithm; HNN: Hopfield neural network; ILP: Integer Linear Programming; NLIP: Non-linear Integer Programming; MILNP: Mixed Integer Non-linear Programming; ML: Machine learning; MVA: Modified Viterbi algorithm; TPO: Two-phase optimization; ICQP: Integer Quadratic Constraint Programming; VLMCE: Model compression based on Virtual Link Mapping Cost Estimation.

5.3.3 SFC placement

In NFV, a requested service is implemented by a set of VNFs that are connected by data flows in a defined order. SFC [95] allows defining an ordered sequence of network functions (NFs) to provide end-to-end services [96]. Typically, a network SFC is designed to deliver data stream services by passing data through a sequence of functions, which can also be shared across multiple applications. For instance, a video stream might pass through rendering, tracking, firewall, and encryption services in software-defined wide-area networks. In NFV, the SFC can be established by efficiently and flexibly placing the constituent VNFs. Traditionally, SFC placement is usually addressed as a code or task offloading [97, 7], also described as a task assignment problem. Nevertheless, the placement of SFC functions presents a rather intricate problem, requiring a solution that not only meets the requirements and demands of each NF but also adheres to various constraints and prerequisites arising from the temporal order and logical dependencies among the NFs.

In [49] authors analyze the SFC placement requirements and limitations of current works. They recommend some useful assumptions that need to be considered as part of the VNF placement problem. The deployment of VNFs poses challenges due to their unique configurations and specific requirements. Existing solutions often overlook the need to describe these specific demands in VNF descriptors, making it difficult to create generalized deployment descriptors. A thorough analysis of related works (see Table 5.2) reveals that none adequately address this requirement. In contrast, our work addresses this requirement by incorporating the particular specificities of VNFs into the problem model.

Our approach recognizes that certain VNFs may require execution on nodes with specific hardware configurations, such as GPUs optimized for deep learning

or specialized CPU setups. Additionally, certain VNFs in the user plane may necessitate special peripherals like video cameras or live streaming equipment. In terms of node resources, our approach encompasses those considered by other approaches, including peripheral resources (fourth column of Table 5.2). This involves not only verifying the presence of necessary peripherals, but also ensuring that their characteristics meet the desired QoS, such as camera resolution.

Furthermore, the implementation environment of each network node significantly influences the selection of deployable VNFs in a specific network slice. Therefore, it is crucial to consider the hardware dependencies of these VNFs when planning VNF chaining to meet users' QoS requirements. Adequately addressing this requirement enables service providers to deploy certain VNFs, such as VoIP-related services, on high-capacity nodes to ensure the desired QoS. However, after reviewing the papers of Table 5.2 we found that none of them appropriately address this requirement (sixth column). Except for our work, previous studies do not allow for flexible resource allocation based on existing resources while ensuring a minimum QoS, nor do they support VNF configuration or reallocation as part of the solution. Only [99, 104] permit VNF reallocation or configuration.

Another essential requirement is minimizing the energy consumption of VNF chains [112] (second column of Table 5.2). However, this aspect is often overlooked in existing works. Although several energy-aware SFC placement approaches have been proposed ([100, 107, 108, 109, 110, 111]), they primarily focus on optimizing energy rates without considering sustainability. To achieve a sustainable VNF placement solution, it is crucial to differentiate nodes that use clean energy from others. Our approach meets this requirement by considering nodes powered by renewable energy as non-consumers, either partially or fully. Consequently, as part of our energy reduction strategy, we prioritize deploying VNFs in nodes where energy savings are expected to be higher.

In the field of SFC placement, a wide range of algorithms is utilized, including ILP variations, heuristics, and machine learning methods (fifth column of Table 5.2). Machine learning approaches are favored due to their ability to avoid scalability issues and provide prompt placement decisions. Reinforcement learning, deep learning, and deep reinforcement learning are commonly adopted techniques. However, recent surveys on VNF placement in edge and cloud computing highlight limitations of machine learning approaches [37, 38]. These limitations include homogeneous scenarios, limited optimization objectives, uncertainty in finding optimal solutions [38], and the time and energy costs associated with training in supervised learning approaches. In contrast, our approach [14] has been tested and found to have negligible energy costs.

Furthermore, with the exception of specific works ([103, 104]), most analyzed solutions in the literature are proprietary, making it challenging to apply them

in different network environments like B5G. In conclusion, our approach offers a valuable and distinctive solution that addresses important requirements discussed in the literature. Moreover, it provides a modular solution implemented on the OSM standard platform.

5.4 Dynamic weaving in smartphones

A dynamic adaptive system is defined by different dimensions or characteristics, such as the main reconfiguration objective (e.g., improve performance, save energy), the type of adaptive engine (application or system-oriented), and the type of the adaptive engine (application or system-oriented) [44, 113], or the context that is monitored (e.g., device status, environment, deployment, user's interactions) [114].

In mobile applications, despite the fact that adaptations are frequently driven by non-functional requirements such as improving the application's performance [44, 115], increasing the failure tolerance [47], or improving the user experience [114, 116, 117], among others; reducing energy consumption has received the most attention in the literature [118, 119, 44, 120, 119, 121].

Specific programming techniques have been used to reconfigure mobile applications, as for example reflection [43], polymorphic methods [44], dynamic proxies [45], Aspect-Oriented Programming [46], and Context-Oriented Programming [47]. Generally, the advantage of employing specific programming techniques lies in their platform independence, thereby enabling their applicability on various mobile operating systems. However, the veracity of this assertion hinges upon the particular technique or technology being utilized.

Chapter 6

Conclusions and Future Work

This chapter presents the conclusions of the thesis and future work.

6.1 Conclusions

The current climate crisis has led to an increased focus on the adoption of sustainable practices in all processes and activities. The main objective is to reduce global warming and address the energy production crisis. The Royal Society reports that information technologies contribute to approximately 1.4% to 5.9% of global greenhouse gas emissions. The way software is designed, developed, and deployed plays a critical role in deciding its energy consumption levels.

Our research focuses on establishing an automated process that effectively decreases energy consumption throughout every phase of software development, ranging from its initial production to its final deployment. It facilitates the energy-aware construction and deployment of software in B5G/Edge/Cloud infrastructures, using a software product line approach that pursues: (i) separate infrastructure and application modeling using SPL feature models; (ii) help infrastructure administrators avoid creating software not supported by the infrastructure and/or know the infrastructure needs to run a specific software product; (iii) optimize software implementations to the characteristics of the hosting infrastructure to reduce energy; (iv) optimize deployment in order to minimize energy consumption and/or latency; (v) perform deployment through an energy-aware orchestrator and (vi) orchestrate infrastructure's resources efficiently; and (vii) reduce the number of functionalities deployed on the user's device by adapting user applications based on contextual information. As a result, our approach has been defined, instantiated, evaluated and applied to multiple case studies and real scenarios.

Our proposal uses the advantages of variability modelling of SPL to consider the characteristics of the infrastructure during the construction of software. By

separating the modeling of software and hardware variability through multi-layer variability models, it enables the consideration of infrastructure constraints on applications. This separation of concerns allows the creation of flexible and reusable infrastructure models in various scenarios. In this thesis, two extensible and reusable B5G/Edge/Cloud infrastructure models (hardware and software) have been presented, specially designed to estimate energy consumption. By means of a set of algorithms and solvers, our approach processes the variability models to avoid the creation of software not supported by the infrastructure, knows the infrastructure needs to run a specific software product and selects the most energy-efficient implementation for each application's functionality, enabling fine-grained optimizations of application to infrastructure. The models, combined with application information, enable the estimation of deployment energy consumption and execution time using physical energy and latency models. Through an energy-aware task allocation framework that uses the information of the configured variability models, our approach provide energy and latency-aware deployments while assuring QoS. The resulting processes and set of tools have been implemented as extension of the Open Source MANO project hosted by the ETSI. It enables the energy-efficient creation, deployment, modification, and orchestration of workloads on B5G/Edge/Cloud infrastructures. Finally, in MEC scenarios, part of the application is deployed on the user device. In this final step, our approach reduces energy by reducing the amount of functionalities deployed on the user's device by adapting the application to the user's behaviour.

Real-world testing has demonstrated the effectiveness of our approach, resulting in significant reductions in energy consumption by up to 59% while assuring QoS, and by up to 51% compared to default Kubernetes deployment. By adapting the application to user behavior, our approach also achieves up to a 20% reduction in energy consumption on users' devices with a negligible energy overhead. Extensive evaluations have been conducted to assess the applicability and scalability of our approach. The results indicate that its implementation in resource-constrained infrastructures is entirely possible, and it has been proven that the additional energy consumption is negligible when compared to the energy savings achieved.

We expect our approach to: (1) help reduce the carbon footprint of IT; (2) facilitates the construction and deployment of energy-aware software regardless of the programmer's prior knowledge; and (3) improve the maintainability ans sustainability of B5G/Edge/Cloud infrastructures.

6.2 Future Work

In future work, an important direction will be to extend the modelling of hardware and software characteristics of the nodes involved in the deployment process.

While the current study has taken into account various factors that impact energy consumption and latency, there is still room for identifying additional factors that may influence these metrics. By thoroughly investigating and incorporating these factors into the models, we can enhance the accuracy and generalizability of our approach.

Another promising avenue for future work lies in leveraging the outputs generated by the optimal models proposed in this thesis to train machine learning models. In order to identify the most energy-efficient deployment options, our approach utilizes solvers, including SMT and MILP, which through a physical energy model provide either the optimal solution or determine if deployment is not possible. By harnessing the power of machine learning, these models can be trained with the results of our optimal solvers and be deployed within an orchestrator to learn from the energy results derived from platforms like Scaphandre. This feedback loop between the machine learning models and the deployment solutions can foster continuous improvement and adaptation in the optimization process. The machine learning models can learn from the energy consumption results and dynamically adjust the deployment decisions to further optimize energy efficiency.

Part II

Thesis Publications

Chapter 7

Supporting IoT applications deployment on edge-based infrastructures using multi-layer feature models

Title:	Supporting IoT applications deployment on edge-based infrastructures using multi-layer feature models
Authors:	Ángel Cañete, Mercedes Amor, Lidia Fuentes
Journal:	Journal of Systems and Software (JSS)
JCR Impact Factor:	2.829 (Q1)
Publication Date:	January 2022
DOI:	https://doi.org/10.1016/j.jss.2021.111086

Abstract. Edge Computing proposes to use the nearby devices in the frontier/Edge of the access network for deploying application tasks of IoT-based systems. However, the functionality of such cyber-physical systems, which is usually distributed in several devices and computers, imposes specific requirements on the infrastructure to run properly. The evolution of an application to meet new user requirements and the high diversity of hardware and software technologies in the IoT/Edge/Cloud can complicate the deployment of continuously evolving applications. The aim of our approach is to apply Multi Layer Feature Models, which capture the variability of applications and the software and hardware infrastructure, to support the deployment in edge-based environments of cyber-physical applications. With this multi-layered approach is possible to support the evolution of application and infrastructure independently. Considering that

IoT/Edge/Cloud infrastructures are usually shared by many applications, the deployment process has to assure that there will be enough resources for all of them, informing developers about the feasible alternatives. We provide four modules so that the developer can calculate what is the configuration of minimal set of devices supporting application requirements of the evolved application. In addition, the developer can find what is the application configuration that can be hosted in the current infrastructure. The successive solutions of continuous deployment generated by our approach pursue the reduction of the system energy footprint and/or execution latency.

Chapter 8

Energy-efficient deployment of IoT applications in edge-based infrastructures: A software product line approach

Title:	Energy-efficient deployment of IoT applications in edge-based infrastructures: A software product line approach
Authors:	Ángel Cañete, Mercedes Amor, Lidia Fuentes
Journal:	IEEE Internet of Things Journal
JCR Impact Factor:	9.936 (Q1)
Publication Date:	January 2021
DOI:	https://doi.org/10.1109/JIOT.2020.3030197

Abstract. In order to lower latency and reduce energy consumption, Edge Computing proposes offloading some computation intensive tasks usually performed in the Cloud onto nearby devices in the frontier/Edge of the access networks. However, current task offloading approaches are often quite simple. They neither consider the high diversity of hardware and software technologies present in edge network devices, nor take into account that some tasks may require some specific software and hardware infrastructure to be executed. This paper proposes a task offloading process that leans on Software Product Line technologies, which are a very good option to model the variability of software and hardware present in edge environments. Firstly, our approach automates the separation of application tasks, considering the data and operation needs and restrictions among them, and identifying the hardware and software resources required by each task. Secondly,

our approach models and manages separately the infrastructure available for task offloading, as a set of nodes that provide certain hardware and software resources. This separation allows to reason about alternative offloading of tasks with different hardware and software resource requirements, in heterogeneous nodes and minimizing energy consumption. In addition, the offloading process considers alternative implementations of tasks to choose the one that best fits the hardware and software characteristics of available edge network infrastructure. The experimental results show that our approach reduces the energy consumption in the user node by approximately 41%-62%, and the energy consumption of the devices involved in a task offloading solution by 34%–48%.

Chapter 9

An NFV solution for energy-efficient placement and resource allocation

Title:	HADES: An NFV solution for energy-efficient placement and resource allocation in heterogeneous infrastructures
Authors:	Ángel Cañete, Mercedes Amor, Lidia Fuentes
Journal:	Journal of Network and Computing Applications (JNCA)
JCR Impact Factor:	8.7 (Q1)
Publication Date:	January 2024
DOI:	https://doi.org/10.1016/j.jnca.2023.103764

Abstract. Network Function Virtualization (NFV) aims to replace traditional network functions running in proprietary hardware with software instances (i.e., Virtual Network Functions, VNFs) embedded in general-purpose virtualization solutions. Aware that the transition to a fully virtualized network infrastructure will pay a high energy cost, especially in IoT systems composed of a myriad of devices, energy efficiency is one of the key innovative targets of future networks. Edge computing should be considered in IoT environments to save time and energy by processing data near the producer devices. However, applying NFV in the context of IoT/Edge/Cloud environments complicates the placement of VNFs, due to the inherent heterogeneous nature of such environments and the variety of resource demands. This paper proposes an energy-aware placement of service function chains of VNFs and a resource-allocation solution for heterogeneous edge infrastructures that considers the computation and communication delays according to the VNFs' location in the infrastructure. The solution has been integrated with

the ETSI-sponsored project Open Source Management and Orchestration (OSM) as an extension called HADES, which allows the configuration of VNFs and their subsequent resource allocation and deployment at the edge, minimizing energy consumption and ensuring a quality of service. We have applied the deployment of augmented reality services in real and simulated scenarios. The results show up to a 59% reduction in power consumption and QoS compliance in all scenarios considered compared to default OSM placement and four other allocation policies. We prove that our solution has negligible power overhead, and validate the scalability and applicability of HADES.

Chapter 10

Energy efficient adaptation engines for android applications

Title:	Energy efficient adaptation engines for android applications
Authors:	Ángel Cañete, José Miguel Horcas, Inmaculada Ayala, Lidia Fuentes
Journal:	Information and Software Technology (IST)
JCR Impact Factor:	2.73 (Q2)
Publication Date:	February 2020
DOI:	https://doi.org/10.1016/j.infsof.2019.106220

Abstract. **Context:** The energy consumption of mobile devices is increasing due to the improvement in their components (e.g., better processors, larger screens). Although the hardware consumes the energy, the software is responsible for managing hardware resources such as the camera software and its functionality, and therefore, affects the energy consumption. Energy consumption not only depends on the installed code, but also on the execution context (environment, devices status) and how the user interacts with the application.
Objective: In order to reduce the energy consumption based on user behavior, it is necessary to dynamically adapt the application. However, the adaptation mechanism also consumes a certain amount of energy in itself, which may lead to an important increase in the energy expenditure of the application in comparison with the benefits of the adaptation. Therefore, this footprint must be measured and compared with the benefit obtained.
Method: In this paper, we (1) determine the benefits, in terms of energy consumption, of dynamically adapting mobile applications, based on user behavior;

and (2) advocate the most energy-efficient adaptation mechanism. We provide four different implementations of a proposed adaptation model and measure their energy consumption.
Results: The proposed adaptation engines do not increase the energy consumption when compared to the benefits of the adaptation, which can reduce the energy consumption by up to 20%.
Conclusion: The adaptation engines proposed in this paper can decrease the energy consumption of the mobile devices based on user behavior. The overhead introduced by the adaptation engines is negligible in comparison with the benefits obtained by the adaptation.

Part III

Appendices

Appendix A

Publications

This chapter presents all publications made in the context of the thesis. The content of such publications is part of the contributions of this thesis, or is directly related to it. In total there are 12 publications over the 5 years of PhD (2018-2023) distributed as follows: 4 JCR-indexed journals, 2 international conferences, 3 workshops in international conferences, and 3 national conferences. Table A.1 lists the publications in journals, Table A.2 list the publications in international conferences, Table A.3 lists the publications in workshops, and Table A.4 lists the publication in national conferences.

A. PUBLICATIONS

Table A.1: Publications on journals.

JOURNALS	
Title:	Energy efficient adaptation engines for android applications
Authors:	Ángel Cañete*, José Miguel Horcas, Inmaculada Ayala, Lidia Fuentes
Journal:	Information and Software Technology (IST)
JCR Impact Factor:	2.73 (Q2)
Publication Date:	February 2020
DOI:	https://doi.org/10.1016/j.infsof.2019.106220
Title:	Energy-efficient deployment of IoT applications in edge-based infrastructures: A software product line approach
Authors:	Ángel Cañete*, Mercedes Amor, Lidia Fuentes
Journal:	IEEE Internet of Things Journal
JCR Impact Factor:	9.936 (Q1)
Publication Date:	January 2021
DOI:	https://doi.org/10.1109/JIOT.2020.3030197
Title:	Supporting IoT applications deployment on edge-based infrastructures using multi-layer feature models
Authors:	Ángel Cañete*, Mercedes Amor, Lidia Fuentes
Journal:	Journal of Systems and Software
JCR Impact Factor:	2.829 (Q1)
Publication Date:	January 2022
DOI:	https://doi.org/10.1016/j.jss.2021.111086
Title:	HADES: An NFV solution for energy-efficient placement and resource allocation in heterogeneous infrastructures
Authors:	Ángel Cañete*, Mercedes Amor, Lidia Fuentes
Journal:	Journal of Network and Computer Applications
JCR Impact Factor:	8.7 (Q1)
Publication Date:	January 2024
DOI:	https://doi.org/10.1016/j.jnca.2023.103764

* Corresponding author.

Table A.2: Publications on international conferences.

INTERNATIONAL CONFERENCES	
Title:	Optimal assignment of augmented reality tasks for edge-based variable infrastructures
Authors:	Ángel Cañete*, Mercedes Amor, Lidia Fuentes
Conference:	13th International Conference on Ubiquitous Computing and Ambient Intelligence (UCAmI)
Conference Rating:	—
Publication Date:	December 2019
DOI:	https://doi.org/10.3390/proceedings2019031028
Title:	A proactive energy-aware auto-scaling solution for edge-based infrastructures
Authors:	Ángel Cañete*, Karim Djemame, Mercedes Amor, Lidia Fuentes, Abdullah Aljulayfi
Conference:	15th International Conference on Utility and Cloud Computing (UCC)
Conference Rating:	—
Publication Date:	December 2022
DOI:	https://doi.org/10.1109/UCC56403.2022.00044

* Corresponding author.

Table A.3: Publications on workshops, tutorials, and doctoral symposium.

WORKSHOPS, TUTORIALS, AND DOCTORAL SYMPOSIUM	
Title:	Energy efficient assignment and deployment of tasks in structurally variable infrastructures (***Doctoral Symposium***)
Authors:	Ángel Cañete*
Conference:	23rd International Systems and Software Product Line Conference (SPLC) - Volume B
Conference Rating:	GGS: A- / CORE:B / LiveSHINE: A, MA: A-
Publication Date:	September 2019
DOI:	https://dl.acm.org/doi/abs/10.1145/3307630.3342704
Title:	Supporting the evolution of applications deployed on edge-based infrastructures using multi-layer feature models
Authors:	Ángel Cañete,* Mercedes Amor, Lidia Fuentes
Conference:	3rd International Workshop on Variability and Evolution of Software-intensive Systems (VariVolution) - 24th International Systems and Software Product Line Conference (SPLC) - Volume B
Conference Rating:	GGS: A- / CORE:B / LiveSHINE: A, MA: A-
Publication Date:	September 2020
DOI:	https://doi.org/10.1145/3382026.3425772
Title:	Energy-Aware Placement of Network Functions in Edge-Based Infrastructures with Open Source MANO and Kubernetes
Authors:	Ángel Cañete,* Alberto Rodríguez, Mercedes Amor, Lidia Fuentes
Conference:	20th International Conference on Service-Oriented Computing (ICSOC)
Conference Rating:	GGS: A- / CORE:A / LiveSHINE: A, MA: B
Publication Date:	December 2022
DOI:	https://doi.org/10.1007/978-3-031-26507-5_15

* Corresponding author.

Table A.4: Publications on national conferences.

NATIONAL CONFERENCES	
Title:	Mecanismos de reconfiguración eco-eficiente de código en aplicaciones móviles Android
Authors:	Ángel Cañete,* Mercedes Amor, Lidia Fuentes
Conference:	XXIII Jornadas de Ingeniería del Software y Base de Datos (JISBD)
Conference Rating:	—
Publication Date:	September 2018
DOI:	https://riuma.uma.es/xmlui/handle/10630/16574
Title:	Sistema de asignación de tareas energéticamente eficiente en infraestructuras de despliegue variables
Authors:	Ángel Cañete,* Mercedes Amor, Lidia Fuentes
Conference:	XXIV Jornadas de Ingeniería del Software y Base de Datos (JISBD)
Conference Rating:	—
Publication Date:	September 2019
DOI:	https://riuma.uma.es/xmlui/handle/10630/18522
Title:	Despliegue Energéticamente Eficiente de Aplicaciones Distribuidas en Infraestructuras en el Borde Heterogéneas
Authors:	Ángel Cañete,* Alberto Rodríguez, Mercedes Amor, Lidia Fuentes
Conference:	XXV Jornadas de Ingeniería del Software y Base de Datos (JISBD)
Conference Rating:	—
Publication Date:	September 2021
DOI:	https://doi.org/10.5281/zenodo.5075287

* Corresponding author.

Appendix B

Resumen en Español

B.1 Introducción

En el mundo actual, existe un énfasis creciente en la adopción de prácticas sostenibles para mitigar el calentamiento global y abordar la crisis de producción de energía. Los proveedores de Tecnología de la Información (TI), los operadores de red y las empresas de telecomunicaciones están promoviendo soluciones de red más eficientes y sostenibles. Las redes móviles Beyond 5G (B5G) se centran en lograr una mayor eficiencia energética, mientras que la tecnología 5G/B5G permite una comunicación más rápida entre dispositivos conectados, promoviendo la expansión de los sistemas IoT.

El IoT y los sistemas ciber-físicos (CPSs por sus siglas en inglés) están cada vez más presentes, generando una gran cantidad de datos que deben ser procesados. La Computación en la Nube ha sido el paradigma dominante para satisfacer estas necesidades, pero su alto consumo de energía y latencia [2] están generando la búsqueda de alternativas como la Computación en el Borde (EC por sus siglas en inglés) [4, 5, 3]. La EC aprovecha la capacidad computacional inactiva de dispositivos en el borde de Internet, reduciendo la cantidad de datos transmitidos a través de Internet y mejorando la eficiencia energética.

Según la socidedad científica Royal Society, las tecnologías de la información son responsables de aproximadamente del 1.4% al 5.9% de las emisiones globales de gases de efecto invernadero. El consumo de energía de las TI está determinado por las prácticas de desarrollo y la utilización de recursos del software. La combinación de infraestructuras de Borde y Nube puede contribuir a reducir la huella energética de los servicios IoT/B5G mientras se mantiene la calidad de servicio (QoS), pero su implementación es compleja debido a la heterogeneidad de dispositivos y tecnologías [4, 7, 6, 8].

El desarrollo de software consciente de la infraestructura puede ayudar a op-

timizar el uso de recursos y la eficiencia energética. Para aliviar la complejidad de este proceso potencialmente desconocido para los desarrolladores, las Líneas de Productos de Software (SPL por sus siglas en inglés) permiten modelar la variabilidad de las características y componentes de un sistema de software. La asignación de tareas a dispositivos debe considerar la latencia, el consumo de energía y la demanda de recursos. La integración de una solución de programación consciente de la energía con una plataforma de orquestación de recursos facilita el despliegue y la gestión de dispositivos y servicios de borde/nube.

La adaptación de funcionalidades de la aplicación a las características de los dispositivos de borde/nube puede reducir el consumo de energía. Además, el comportamiento del usuario y el contexto son importantes para determinar qué tareas deben ser desplegadas en el dispositivo del usuario, permitiendo adaptar las funcionalidades desplegadas en función de los mismos.

En esta tesis, se propone un marco de solución completo energéticamente eficiente que aborda desde la creación de software hasta el despliegue consciente de la energía en infraestructuras B5G, del borde y la nube. Se han desarrollado distintas soluciones para el desarrollo de software distribuido consciente de la energía y una solución de asignación de tareas energéticamente eficiente que asegura una QoS específica. El conjunto de soluciones resultante se han integrado como una extensión del proyecto Open Source MANO para la orquestación eficiente en energía de cargas de trabajo en infraestructuras B5G, del borde y la nube. Por ulitmo, se han elaborado varios motores de adaptación de aplicaciones móviles energéticamente eficientes y transparentes para el usuario. Las soluciones llevadas a cabo en el marco de la presente tesis se han evaluado ampliamente en escenarios reales, validando su aplicabilidad, escabilidad, y bajo consumo energético respecto al ahorro que supone su implantación.

B.2 Antecedentes

Esta tesis presenta los antecedentes de las SPL, especialmente de los modelos de características. También introduce los solucionadores (*solvers* en inglés), sus tipos y caracterísitcas. Asimismo, ofrece una visión general de otros aspectos como el consumo de energía en TI, la virtualización, computación en el borde, descarga de código, orquestación de cargas de trabajo, programación de tareas, auto-escalado de recursos y el despliegue de características centrado en el usuario. Estas tecnologías se utilizan para gestionar la variabilidad, optimizar el consumo de energía, desvincular recursos de hardware y software, procesar datos en el borde de la red, mejorar la experiencia del usuario, asignar tareas computacionales, optimizar la asignación de recursos y maximizar el rendimiento en sistemas de computación en el borde.

El paradigma clásico de la ingeniería de SPL separa dos procesos principales: la Ingeniería de Dominio y la Ingeniería de Aplicación. La Ingeniería de Dominio es el proceso de analizar el área de conocimiento de una SPL y desarrollar artefactos reutilizables, mientras que la Ingeniería de Aplicación es el proceso de desarrollar un producto específico para las necesidades de un cliente. Las características son representaciones abstractas del dominio, mientras que el espacio de la solución se refiere a las perspectivas de los desarrolladores y proveedores y a las actividades relacionadas con el diseño, la implementación y la verificación de características.

En la Ingeniería de Dominio se realiza el modelado de la variabilidad mediante modelos de características que permiten especificar los elementos comunes y variables del sistema. Los modelos de características (*Feature Models* en inglés (FMs)) se representan como un conjunto de características jerárquicas y restricciones cruzadas que representan las relaciones entre ellas. Además de la variabilidad del dominio, una SPL puede tener variabilidad independiente del dominio y variabilidad interna. Para gestionar estas dimensiones adicionales de variabilidad, se utilizan modelos de características multicapa con restricciones cruzadas entre modelos.

Los solucionadores son herramientas de software que resuelven problemas específicos. En el contexto de la ingeniería de SPL, los solucionadores juegan un papel crucial en el modelado de variabilidad al permitir un análisis eficiente y un razonamiento automatizado, siendo capaces de manejar problemas de satisfacción de restricciones y encontrar soluciones óptimas o casi óptimas basadas en objetivos y restricciones definidos.

La computación en el borde (*Edge Computing* (EC) en inglés) es un paradigma de computación distribuida que promueve el procesamiento de datos en el borde de la red, más cerca de las fuentes de datos. Esto es especialmente importante en el contexto del IoT y los CPSs, donde hay un gran número de dispositivos interconectados generando datos a gran escala. La computación en el borde móvil (*Mobile Edge Computing* (MEC) en inglés) es una aplicación específica de la computación en el borde, centrada mejorar la experiencia del usuario al reducir la latencia y la carga de la red del dispositivo móvil del usuario. La descarga de código es una técnica utilizada en MEC para migrar tareas computacionalmente intensivas a sistemas externos y mejorar el rendimiento y la vida útil de la batería de los dispositivos móviles.

Una tecnología clave en EC es la virtualización, que permite desvincular los recursos hardware del software y la ejecución de múltiples instancias en el mismo hardware mediante el uso de software que imita las características del hardware. Una solución de virtualización ampliamente utilizada son los contenedores, que permiten encapsular aplicaciones y sus dependencias en entornos aislados y portátiles.

Para gestionar los recursos de estos entornos virtualizados se emplean orquesta-

dores, que permiten definir el comportamiento de los contenedores y automatizan el despliegue, la gestión y el escalado de aplicaciones basadas en contenedores. Una de las funciones principales de los orquestadores es la asignación de tareas a los nodos de la infraestructura gestionada. La asignación de tareas es un problema complejo debido a la naturaleza distribuida de la infraestructura y la heterogeneidad de las tareas y los nodos. Existen varios métodos para resolver problemas de programación de tareas, como solvers, algoritmos heurísticos y técnicas de aprendizaje automático. Los solvers ofrecen soluciones óptimas, mientras que los algoritmos heurísticos y las técnicas de aprendizaje automático proporcionan soluciones subóptimas.

Por otro lado, hay momentos en que algunos nodos no tienen cargas de trabajo asignados. El auto-escalado de recursos es un método utilizado para asignar dinámicamente recursos según las necesidades del sistema y maximizar el rendimiento mientras se reducen los costos y el consumo energético. El auto-escalado se puede realizar aumentando/disminuyendo el número de nodos o modificando los recursos disponibles dentro de los nodos.

Por úlitimo, en entornos MEC parte de las cargas de trabajo se despliegan en el dispositivo del usuario. En estos casos es posible disminuir la cantidad de cargas de trabajo desplegadas en el mismo mediante un despliegue consciente del contexto. Esto permite analizar el comportamiento del usuario y evitar el despliegue de características no utilizadas o irrelevantes en el dispositivo móvil, lo que reduce el consumo de energía.

B.3 Motivación

Las aplicaciones B5G/IoT/Edge/Cloud son altamente variables debido a la diversidad de dispositivos, sensores y plataformas utilizados. Esto implica una multitud de combinaciones y configuraciones posibles. Abordar esta variabilidad es crucial para desarrollar software compatible con la infraestructura IoT/B5G. Utilizar un enfoque basado en SPL permitiría garantizar la satisfacción de las dependencias de la aplicación y la infraestructura, y asegurar que haya suficientes recursos para atender las demandas de tareas de la aplicación. Por lo tanto, la primera pregunta clave (acrónimo inglés **RQ1**) es: **¿Cómo podemos modelar la gran variabilidad hardware y software de las infraestructuras B5G/Edge/Cloud centrándonos en el consumo de energía? ¿Es factible garantizar la satisfacción de las dependencias entre aplicaciones e infraestructuras y la disponibilidad de recursos para la demanda de las aplicaciones?**

Una vez configurada la infraestructura y la aplicación deseada, las implementaciones de características seleccionadas pueden variar en eficiencia energética. Es importante evaluar y seleccionar la opción más apropiada considerando las carac-

terísticas de la infraestructura. Tras esto, los desarrolladores enfrentan el desafío de asignar tareas mientras aseguran un nivel específico de calidad de servicio (QoS en inglés) y optimizar el consumo de energía. El problema de asignación de tareas a nodos es muy complejo y costoso computacionalmente, por lo que es necesario encontrar soluciones de asignación válidas y eficientes en términos de energía. Así, se plantea **RQ2: ¿Cómo podemos seleccionar las implementaciones más eficientes en términos de energía que se alineen con el estado de la infraestructura y aseguren la QoS? ¿Cómo podemos automatizar el despliegue de funciones de software en infraestructuras B5G/Edge/Cloud?**

Aunque abordar RQ1 y RQ2 podría mejorar la creación y despliegue de software consciente de la energía en infraestructuras B5G/Edge/Cloud, su aplicabilidad en diferentes escenarios sigue siendo un desafío. La mayoría de las soluciones de asignación de tareas son propietarias y no se pueden reutilizar en diferentes entornos de red como B5G. En las redes B5G, los orquestadores supervisan los servicios contenerizados, pero no consideran el consumo de energía ni garantizan una calidad de servicio específica. ETSI desarrolló un marco de arquitectura de referencia para la gestión y orquestación de NFV (MANO), pero no establece estándares para su implementación. La falta de técnicas de auto-escalado diseñadas específicamente para infraestructuras de borde también es un desafío. Por lo tanto, se plantea **RQ3: ¿Cómo podemos integrar la conciencia energética en las técnicas de orquestación de recursos para garantizar la QoS y lograr despliegues energéticamente eficientes en las redes de próxima generación?**

En MEC, se puede reducir el consumo de energía al omitir el despliegue de funcionalidades no utilizadas o irrelevantes para las necesidades del usuario en su smartphone. Esto se logra analizando los patrones y preferencias del usuario, así como el estado del dispositivo, para adaptar dinámicamente la aplicación y ejecutar solo los componentes esenciales. Esta adaptación debe ser energéticamente eficiente y transparente para el usuario. El problema de la movilidad también es un desafío en los escenarios de MEC, y tener la aplicación completa almacenada en el dispositivo del usuario garantiza una experiencia ininterrumpida incluso en caso de pérdida de conectividad o cambio de dispositivos. Por lo tanto, definimos **RQ4: ¿Qué métodos se pueden utilizar para adaptar dinámicamente las aplicaciones de los usuarios basándose en la información contextual, manteniendo la transparencia para el usuario y optimizando la eficiencia energética?**

B.4 Propuesta

Nuestra propuesta proporciona una visión general de los enfoques incrementales utilizados para abordar las cuatro RQs y optimizar el consumo de energía en

infraestructuras y aplicaciones de IoT/Edge/Cloud/B5G. Consiste en cuatro procesos principales: Ingeniería de Dominio, Ingeniería de Configuración, Ingeniería de Aplicaciones y Ingeniería de Despliegue.

En el proceso de Ingeniería de Dominio, la variabilidad de las infraestructuras y aplicaciones de IoT/Edge/Cloud/B5G se modela utilizando FMs. Se utilizan FMs multicapa, FMs con cardinalidad y atributos numéricos para capturar y representar las variabilidades dentro del sistema. En la presente tesis se presentan dos FMs extensibles y reusables que representan la variabilidad hardware y software de las infraestructuras B5G/Edge/Cloud [12].

El proceso de Ingeniería de Configuración se centra en analizar los requisitos funcionales y no funcionales de la infraestructura y las restricciones impuestas a las aplicaciones por la infraestructura. Se presentan dos módulos, el *Application Variability Adapter* (AVA) y el *New Devices Finder* (NDF) [12], para encontrar la configuración óptima e identificar los dispositivos adicionales necesarios para soportar las aplicaciones. Ambos utilizan un solver para obtener la solución. Su escalabilidad y aplicabilidad en entornos reales ha sido ampliamente evaluada.

En el proceso de Ingeniería de Aplicaciones, los FMs de la aplicación e infraestructura se encuentran configurados. En este paso, se asignan las funcionalidades de la aplicación seleccionadas con las interfaces de tarea apropiadas, y mediante el módulo *Tasks Implementation Selector* (TIS) [13] se evalúan diferentes implementaciones de tareas para seleccionar la implementación más eficiente en términos de energía para cada funcionalidad de la aplicación.

El proceso de Ingeniería de Despliegue aborda el despliegue de la aplicación en la infraestructura de manera energéticamente eficiente, teniendo en cuenta las asignaciones de tareas optimizadas en el paso anterior y los recursos disponibles. Para ello, se han desarrollado varios módulos de asignación de tareas, capaces de minimizar la latencia y/o consumo energético, asegurando los requisitos funcionales y una cierta QoS en todos los casos. Los módulos de despliegue se han integrado con tecnologías de contenerización y orquestación como Docker y Kubernetes para proporcionar alta disponibilidad y tolerancia a fallos, proponiéndose una solución de auto-escalado [78] para gestionar las necesidades dinámicas de las solicitudes de servicio de los usuarios con el objetivo de minimizar el consumo energético de los nodos inactivos. Como resultado se ha presentado HADES [14], una extensión del proyecto OSM [53] que amplía sus capacidades minimizando el consumo de energía y garantizando la QoS. La arquitectura HADES sigue el bucle N-MAPE-K, una extensión del bucle clásico Monitor-Analyze-Plan-Execute sobre un Conocimiento compartido (MAPE-K, [70]) adaptado para redes, definido en el proyecto DAEMON [71]. Las soluciones de asignación energéticamente eficiente utilizan solvers para dar un resultado, y su aplicabilidad y escalabilidad en entornos reales.

Además, en entornos MEC la funcionalidad de la aplicación desplegada en el

dispositivo móvil del usuario puede ser adaptada basándose en el comportamiento del propio usuario para reducir el consumo de energía. En la tesis se presentan diferentes motores de adaptación para dispositivos inteligentes [15] capaces de modificar la aplicación en tiempo de ejecución o en el momento en que comienza, dependiendo del motor empleado. Nuestros resultados muestran que los motores de adaptación reducen el consumo de energía hasta en un 20% al adaptar la aplicación al comportamiento del usuario, y que su uso aumenta el consumo energético en un 2,51% en el peor de los casos, lo que la convierte en una solución energéticamente eficiente para entornos MEC.

B.5 Discusión de Resultados

En [12] se definen dos modelos de variabilidad extensibles y reutilizables para infraestructuras B5G/Edge/Cloud, diseñados para estimar el consumo de energía. Utilizamos FMs de múltiples capas para lidiar con la alta variabilidad en las aplicaciones e infraestructuras de implementación, y utilizamos restricciones de modelo cruzado para mantener la coherencia entre los FMs de diferentes capas. Los módulos NDF y AVA ayudan a los desarrolladores durante la configuración y posterior implementación de aplicaciones distribuidas utilizando SPLs. Estos módulos analizan y consideran las dependencias entre las aplicaciones e infraestructuras para garantizar que hay suficientes recursos disponibles para satisfacer las demandas de las tareas de la aplicación. El módulo AVA adapta la variabilidad de la aplicación de acuerdo con las capacidades de la infraestructura, mientras que el módulo NDF informa sobre los nuevos dispositivos necesarios para ejecutar una aplicación específica en una infraestructura específica. Por lo tanto, **la respuesta a RQ1 es que utilizando FMs de múltiples capas, podemos modelar la extensa variabilidad de las infraestructuras B5G/Edge/Cloud, con un enfoque específico en características relacionadas con el consumo de energía. A través de nuestros módulos AVA y NDF, abordamos las dependencias de la aplicación y la infraestructura, asegurando suficientes recursos para las tareas de la aplicación.**

El módulo TIS utiliza la información de los FMs configurados para seleccionar la implementación más adecuada, desde el punto de vista energético, para cada funcionalidad de la aplicación. Luego, un modelo de asignación de tareas óptimo asigna cada tarea al nodo más adecuado para minimizar el consumo de energía y garantizar la QoS de la aplicación. Esta solución es aplicable a cualquier aplicación e infraestructura y utiliza modelos de energía y latencia para calcular el consumo de energía y el tiempo de ejecución. Hemos presentado varias versiones del módulo de asignación de tareas óptimo, utilizando diferentes modelos y solucionadores. Los resultados demuestran reducciones en el consumo de energía de

hasta un 62% en el dispositivo del usuario y hasta un 48% en el resto de nodos de la red, y validan la aplicabilidad de la propuesta. Así, **la respuesta a RQ2 es que el módulo TIS permite optimizar la implementación de tareas en la infraestructura para reducir el consumo de energía. Utilizando la descripción de tareas e infraestructura de los modelos de variabilidad, se pueden aplicar modelos de latencia y energía independientes de la aplicación y la infraestructura, automatizando el despliegue de funciones de software en infraestructuras B5G/Edge/Cloud.**

La solución de autoescalado horizontal para infraestructuras heterogéneas propuesta reduce el consumo de energía y es aplicable en diferentes contextos. Esta solución consta de un predictor de carga de trabajo, que utiliza aprendizaje automático para prever la carga esperada, y un módulo que decide qué nodos deben mantenerse activos para satisfacer dicha carga esperada y minimizar el consumo de energía. Esto, junto con la extensión de OSM HADES, permite gestionar infraestructuras B5G/Edge/Cloud de manera energéticamente eficiente requisitos como la memoria y el almacenamiento. Hemos evaluado extensamente HADES y ha demostrado una reducción del consumo de energía de hasta un 59%, siendo de 51% en comparación con el despliegue predeterminado de OSM. Nuestra solución ha demostrado ser aplicable en entornos reales, incluso en computadoras estándar. Por lo tanto, **la respuesta a RQ3 es que la asignación de recursos energéticamente eficientes puede integrarse en técnicas de orquestación existentes, como OSM. Utilizando un modelo de latencia, nuestro descriptor de despliegue puede prever los recursos necesarios en cada nodo B5G/Edge/Cloud para garantizar una QoS específica.**

Hemos desarrollado un esquema de adaptación genérico inspirado en el bucle MAPE-K junto con cuatro motores de adaptación diseñados para aplicaciones Android. Estos motores ofrecen diferentes enfoques para adaptar la aplicación en términos de cómo y cuándo ocurre la adaptación. Hemos demostrado que estos motores tienen un consumo energético mínimo comparado con las reducciones que obtienen. **La respuesta a RQ4 es que mediante un esquema de adaptación basado en el bucle MAPE-K, podemos adaptar dinámicamente las aplicaciones según la información contextual. Se han desarrollado cuatro motores de adaptación diferentes que funcionan de manera transparente para el usuario y tienen una sobrecarga energética mínima.**

B.6 Conclusiones y Trabajo Futuro

Nuestra investigación se ha centrado en reducir el consumo de energía en todas las fases del desarrollo de software, desde la producción hasta el despliegue final. Utilizamos la modelización de la variabilidad de SPL para considerar las carac-

terísticas de la infraestructura durante la construcción del software, asegurando que el software puede ser desplegado. Hemos desarrollado un módulo de asignación de tareas consciente de la energía que usa modelos físicos de energía y latencia para estimar el consumo de energía y el tiempo de ejecución del despliegue. Este conjunto de resultados se ha desarrollado como una extensión del proyecto de código abierto OSM.

Nuestras soluciones se han evaluado ampliamente en entornos reales, demostrando la efectividad de nuestro enfoque, logrando reducciones significativas en el consumo de energía, asegurando la QoS y adaptando la aplicación al comportamiento del usuario. Hemos evaluado la aplicabilidad y escalabilidad de nuestro enfoque, y los resultados indican que su implementación en infraestructuras con recursos limitados es totalmente posible.

Como trabajo futuro, planeamos extender la modelización de las características de hardware y software de los nodos involucrados en el proceso de despliegue, y explorar el uso de modelos de aprendizaje automático para optimizar aún más la eficiencia energética en el despliegue de software.

References

[1] R. Atat, L. Liu, J. Wu, G. Li, C. Ye, and Y. Yang, "Big data meet cyber-physical systems: A panoramic survey," *IEEE Access*, vol. 6, pp. 73 603–73 636, 2018. 3

[2] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, Jan 2017. 4, 69

[3] H. Elazhary, "Internet of Things (IoT), mobile cloud, cloudlet, mobile iot, iot cloud, fog, mobile edge, and edge emerging computing paradigms: Disambiguation and research directions," *Journal of Network and Computer Applications*, vol. 128, pp. 105–140, Nov 2018. 4, 13, 69

[4] S. Bagchi, M.-B. Siddiqui, P. Wood, and H. Zhang, "Dependability in edge computing," *Commun. ACM*, vol. 63, no. 1, p. 58–66, Dec. 2020. 4, 69

[5] G. Premsankar, M. Di Francesco, and T. Taleb, "Edge computing for the internet of things: A case study," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 1275–1284, April 2018. 4, 5, 69

[6] F. Liu, G. Tang, Y. Li, Z. Cai, X. Zhang, and T. Zhou, "A survey on edge computing systems and tools," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1537–1562, 2019. 4, 6, 43, 69

[7] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017. 4, 5, 6, 15, 16, 20, 25, 27, 28, 32, 42, 43, 45, 46, 69

[8] M. Huang, W. Liu, T. Wang, A. Liu, and S. Zhang, "A cloud–MEC collaborative task offloading scheme with service orchestration," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 5792–5805, 2020. 4, 6, 43, 69

[9] V. G. Plc, "Sustainable business report 2019," https://www.vodafone.com/content/dam/vodcom/sustainability/pdfs/sustainablebusiness2019.pdf, online; acc. 3 June 2022. 4, 14

[10] P. Porambage, J. Okwuibe, M. Liyanage, M. Ylianttila, and T. Taleb, "Survey on multi-access edge computing for internet of things realization," *IEEE Communications Surveys Tutorials*, vol. 20, no. 4, pp. 2961–2991, 2018. 5

[11] S. Melendez and M. P. McGarry, "Computation offloading decisions for reducing completion time," in *2017 14th IEEE Annual Consumer Communications Networking Conference (CCNC)*, Jan 2017, pp. 160–164. 5

[12] A. Cañete, M. Amor, and L. Fuentes, "Supporting iot applications deployment on edge-based infrastructures using multi-layer feature models," *Journal of Systems and Software*, vol. 183, p. 111086, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121221001837 6, 11, 28, 35, 36, 74, 75

[13] ——, "Energy-efficient deployment of iot applications in edge-based infrastructures: A software product line approach," *IEEE Internet of Things Journal*, vol. 8, no. 22, pp. 16 427–16 439, 2021. 7, 28, 31, 35, 37, 43, 44, 74

[14] ——, "Hades: An nfv solution for energy-efficient placement and resource allocation in heterogeneous infrastructures," *Journal of Network and Computer Applications*, p. 103764, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804523001832 7, 28, 29, 38, 47, 74

[15] A. Cañete, J.-M. Horcas, I. Ayala, and L. Fuentes, "Energy efficient adaptation engines for android applications," *Information and Software Technology*, vol. 118, p. 106220, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0950584919302307 7, 39, 75

[16] K. Pohl, G. Böckle, and F. J. v. d. Linden, *Software Product Line Engineering: Foundations, Principles and Techniques.* Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2005. 9, 20

[17] K. Kang, S. Cohen, J. Hess, W. Novak, and A. Peterson, "Feature-Oriented Domain Analysis (FODA) feasibility study," Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, Technical Report CMU/SEI-90-TR-021, 1990. 10

[18] J. van Gurp, J. Bosch, and M. Svahnberg, "On the notion of variability in software product lines," in *Proceedings Working IEEE/IFIP Conference on Software Architecture*, 2001, pp. 45–54. 10

[19] K. C. Kang, S. Kim, J. Lee, K. Kim, E. Shin, and M. Huh, "Form: A feature-;oriented reuse method with domain-;specific reference architectures," *Annals of Software Engineering*, vol. 5, pp. 143–168, 1998. 10

[20] J. A. Galindo, D. Dhungana, R. Rabiser, D. Benavides, G. Botterweck, and P. Grünbacher, "Supporting distributed product configuration by integrating heterogeneous variability modeling approaches," *Information and Software Technology*, vol. 62, pp. 78 – 100, 2015. 10

[21] M. Lettner, J. Rodas, J. A. Galindo, and D. Benavides, "Automated analysis of two-layered feature models with feature attributes," *Journal of Computer Languages*, vol. 51, pp. 154 – 172, 2019. 10, 42

[22] D. Batory, "Feature models, grammars, and propositional formulas," in *Software Product Lines*, H. Obbink and K. Pohl, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 7–20. 11

[23] C. Barrett and C. Tinelli, *Satisfiability Modulo Theories*. Cham: Springer International Publishing, 2018, pp. 305–343. [Online]. Available: https://doi.org/10.1007/978-3-319-10575-8_11 11

[24] F. Rossi, P. Van Beek, and T. Walsh, *Handbook of constraint programming*. Elsevier, 2006. 11

[25] R. P. Goldberg, "Survey of virtual machine research," *Computer*, vol. 7, no. 6, pp. 34–45, 1974. 12

[26] Y. Mansouri and M. A. Babar, "A review of edge computing: Features and resource virtualization," *Journal of Parallel and Distributed Computing*, vol. 150, pp. 155–183, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0743731520304317 12, 13, 25

[27] P. Sharma, L. Chaufournier, P. Shenoy, and Y. C. Tay, "Containers and virtual machines at scale: A comparative study," in *Proceedings of the 17th International Middleware Conference*, ser. Middleware '16. New York, NY, USA: Association for Computing Machinery, 2016. [Online]. Available: https://doi.org/10.1145/2988336.2988337 13

[28] A. Madhavapeddy, R. Mortier, C. Rotsos, D. Scott, B. Singh, T. Gazagnaire, S. Smith, S. Hand, and J. Crowcroft, "Unikernels: Library operating systems for the cloud," in *Proceedings of the Eighteenth International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '13. New York, NY, USA: Association

for Computing Machinery, 2013, p. 461–472. [Online]. Available: https://doi.org/10.1145/2451116.2451167 13

[29] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, p. 50–58, apr 2010. [Online]. Available: https://doi.org/10.1145/1721654.1721672 13

[30] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017. 13, 16

[31] G. L. Santos, D. d. F. Bezerra, É. d. S. Rocha, L. Ferreira, A. L. C. Moreira, G. E. Gonçalves, M. V. Marquezini, Á. Recse, A. Mehta, J. Kelner, D. Sadok, and P. T. Endo, "Service function chain placement in distributed scenarios: A systematic review," *Journal of Network and Systems Management*, vol. 30, no. 1, p. 4, Sep 2021. [Online]. Available: https://doi.org/10.1007/s10922-021-09626-4 14

[32] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787 – 2805, 2010. 14

[33] E. A. Lee, "Cyber physical systems: Design challenges," in *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*, May 2008, pp. 363–369. 14

[34] E. Casalicchio, *Container Orchestration: A Survey*. Springer International Publishing, 2019, pp. 221–235. 15, 21

[35] L. Özbakir, A. Baykasoğlu, and P. Tapkan, "Bees algorithm for generalized assignment problem," *Applied Mathematics and Computation*, vol. 215, no. 11, pp. 3782 – 3795, 2010. 16, 21, 29

[36] T. Koch, T. Berthold, J. Pedersen, and C. Vanaret, "Progress in mathematical programming solvers from 2001 to 2020," *EURO Journal on Computational Optimization*, vol. 10, p. 100031, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2192440622000077 16, 28, 37

[37] W. Attaoui, E. Sabir, H. Elbiaze, and M. Guizani, "Vnf and cnf placement in 5g: Recent advances and future trends," *IEEE Transactions on Network and Service Management*, pp. 1–1, 2023. 16, 47

[38] S. Yang, F. Li, S. Trajanovski, R. Yahyapour, and X. Fu, "Recent advances of resource allocation in network function virtualization," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 2, pp. 295–314, Feb 2021. 16, 47

[39] A. Goli., N. Mahmoudi., H. Khazaei., and O. Ardakanian., "A holistic machine learning-based autoscaling approach for microservice applications," in *Proceedings of the 11th International Conference on Cloud Computing and Services Science - CLOSER,*, INSTICC. SciTePress, 2021, pp. 190–198. 16

[40] M. Ganeshalingam, A. Shehabi, and L. Desroches, "Shining a light on small data centers in the u.s." 2017. 16

[41] T. Chen, R. Bahsoon, and X. Yao, "A survey and taxonomy of self-aware and self-adaptive cloud autoscaling systems," *ACM Comput. Surv.*, vol. 51, no. 3, jun 2018. [Online]. Available: https://doi.org/10.1145/3190507 17, 21, 31, 32, 44

[42] M. S. Aslanpour, S. S. Gill, and A. N. Toosi, "Performance evaluation metrics for cloud, fog and edge computing: A review, taxonomy, benchmarks and standards for future research," *Internet of Things*, vol. 12, p. 100273, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2542660520301062 17, 21, 31, 32, 44, 45

[43] J. C. Casquina, J. D. A. S. Eleuterio, and C. M. F. Rubira, "Adaptive deployment infrastructure for Android applications," in *12th European Dependable Computing Conference (EDCC)*, 2016, pp. 218–228. 17, 22, 48

[44] S. Elmalaki, L. F. Wanner, and M. B. Srivastava, "CAreDroid: Adaptation framework for Android context-aware applications," *GetMobile*, vol. 20, no. 2, pp. 35–38, 2016. 17, 22, 48

[45] H. Artail, K. Fawaz, and A. Ghandour, "A proxy-based architecture for dynamic discovery and invocation of web services from mobile devices," *IEEE Transactions on Services Computing*, vol. 5, no. 1, pp. 99–115, 2012. 17, 22, 48

[46] Y. Falcone and S. Currea, "Weave droid: Aspect-oriented programming on Android devices: Fully embedded or in the cloud," in *International Conference on Automated Software Engineering*, ser. ASE, 2012, pp. 350–353. 17, 22, 48

[47] C. Schuster, M. Appeltauer, and R. Hirschfeld, "Context-oriented programming for mobile devices: JCop on Android," in *3rd International Workshop on Context-Oriented Programming*, ser. COP, 2011, pp. 5:1–5:5. 17, 22, 48

[48] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. S. Quek, "Offloading in mobile edge computing: Task allocation and computational frequency scaling," *IEEE Transactions on Communications*, vol. 65, no. 8, pp. 3571–3584, Aug 2017. 20, 28, 43

[49] H. U. Adoga and D. P. Pezaros, "Network function virtualization and service function chaining frameworks: A comprehensive review of requirements, objectives, implementations, and open research challenges," *Future Internet*, vol. 14, no. 2, 2022. [Online]. Available: https://www.mdpi.com/1999-5903/14/2/59 20, 46

[50] "Etsi - standards for nfv," https://www.etsi.org/technologies/nfv, online; accessed 15 February 2022. 21, 45

[51] B. Yi, X. Wang, K. Li, S. K. Das, and M. Huang, "A comprehensive survey of network function virtualization," *Comput. Networks*, vol. 133, pp. 212–262, 2018. 21, 45

[52] L. Mamushiane, A. A. Lysko, T. Mukute, J. Mwangama, and Z. D. Toit, "Overview of 9 open-source resource orchestrating etsi mano compliant implementations: A brief survey," in *2019 IEEE 2nd Wireless Africa Conference (WAC)*, 2019, pp. 1–7. 21, 45

[53] A. Reid, A. González, A. Armengol, G. de Blas, M. Xie, P. Grønsund, P. Willis, P. Eardley, and F. Salguero, "Osm scope, functionality, operation and integration guidelines," *ETSI, White Paper*, 2019. 21, 29, 45, 74

[54] "Opnfv's docs," https://docs.opnfv.org/, online; acc. 3 June 2022. 21, 45

[55] "Onap's docs," https://docs.onap.org/, online; acc. 3 June 2022. 21, 45

[56] "Cloudfy docs," 2023, online; accessed 20 October 2023. [Online]. Available: https://docs.cloudify.co/latest/ 21, 45

[57] S. H. H. Madni, M. S. A. Latiff, Y. Coulibaly, and S. M. Abdulhamid, "Recent advancements in resource allocation techniques for cloud computing environment: a systematic review," *Cluster Computing*, vol. 20, no. 3, pp. 2489–2533, Sep 2017. [Online]. Available: https://doi.org/10.1007/s10586-016-0684-4 21, 44

[58] S. Islam, J. Keung, K. Lee, and A. Liu, "Empirical prediction models for adaptive resource provisioning in the cloud," *Future Generation Computer Systems*, vol. 28, no. 1, pp. 155–162, 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X11001129 21, 44

[59] M. Rosenmüller, N. Siegmund, T. Thüm, and G. Saake, "Multi-dimensional variability modeling," in *Fifth International Workshop on Variability Modelling of Software-Intensive Systems, Namur, Belgium, January 27-29, 2011. Proceedings*, ser. ACM International Conference Proceedings Series. ACM, 2011, pp. 11–20. 23, 35

[60] K. Czarnecki, S. Helsen, and U. Eisenecker, "Formalizing cardinality-based feature models and their specialization," *Software Process: Improvement and Practice*, vol. 10, no. 1, pp. 7–29, 2005. 23, 35

[61] D. Benavides, P. Trinidad, and A. Ruiz-Cortés, "Automated reasoning on feature models," in *Advanced Information Systems Engineering*, ser. LNCS. Springer Berlin Heidelberg, 2005, vol. 3520, pp. 491–503. [Online]. Available: http://dx.doi.org/10.1007/11431855_34 23, 35

[62] W. Zhang, Y. Wen, and D. O. Wu, "Energy-efficient scheduling policy for collaborative execution in mobile cloud computing," in *2013 Proceedings IEEE INFOCOM*, April 2013, pp. 190–194. 25

[63] L. De Moura and N. Bjørner, "Z3: An efficient smt solver," in *Proceedings of the Theory and Practice of Software, 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, ser. TACAS'08/ETAPS'08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 337–340. 26, 29

[64] A. Niewiadomski, J. Skaruz, W. Penczek, M. Szreter, and M. Jarocki, "Smt versus genetic and openopt algorithms: Concrete planning in the planics framework," *Fundamenta Informaticae*, vol. 135, pp. 451–466, 01 2014. 26

[65] N. Bjørner, A.-D. Phan, and L. Fleckenstein, "νz - an optimizing SMT solver," in *Tools and Algorithms for the Construction and Analysis of Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 194–199. 26

[66] A. Cañete, M. Amor, and L. Fuentes, "Optimal assignment of augmented reality tasks for edge-based variable infrastructures," in *13th Int. Conf. on Ubiquitous Computing and Ambient Intelligence, UCAmI 2019, Toledo,*

Spain, December 2-5, 2019, ser. MDPI Proceedings, vol. 31. MDPI, 2019, p. 28. 28, 35

[67] D. Machado, "A benchmark of optimization solvers for genome-scale metabolic modeling," *bioRxiv*, 2023. 28

[68] C. Sundermann, T. Thüm, and I. Schaefer, "Evaluating #sat solvers on industrial feature models," in *Proceedings of the 14th Int. Conference on Variability Modelling of Software-Intensive Systems*, ser. VAMOS '20. New York, NY, USA: ACM, 2020. 29

[69] "Kubernetes' docs," 2023, online; acc. 10 October 2023. [Online]. Available: https://kubernetes.io/docs/home/ 29

[70] O. Gheibi, D. Weyns, and F. Quin, "Applying machine learning in self-adaptive systems: A systematic literature review," *ACM Trans. Auton. Adapt. Syst.*, vol. 15, no. 3, aug 2021. 30, 37, 74

[71] L. Fuentes, M. Amor, and Ángel Cañete et al., "DAEMON Deliverable 4.2: Refined design of intelligent orchestration and management mechanisms," Nov. 2022, This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No. 101017109. 30, 37, 74

[72] K. Kang, S. Cohen, J. Hess, W. Novak, and A. Peterson, "Feature-oriented domain analysis (foda) feasibility study," Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU/SEI-90-TR-021, 1990. [Online]. Available: http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=11231 31

[73] S. Telecom, "The telecom dataset," 2018, online; accessed 20 October 2023. [Online]. Available: http://sguangwang.com/TelecomDataset.html 32

[74] Sonmez et al., "Edgecloudsim: An environment for performance evaluation of edge computing systems," *Transactions on Emerging Telecommunications Technologies*, vol. 29, no. 11, p. e3493. 32

[75] P. Arcaini, E. Riccobene, and P. Scandurra, "Modeling and analyzing MAPE-K feedback loops for self-adaptation," in *Software Engineering for Adaptive and Self-Managing Systems*, ser. SEAMS, 2015, pp. 13–23. 32

[76] J. Lopez, L. Babun, H. Aksu, and A. S. Uluagac, "A survey on function and system call hooking approaches," *Journal of Hardware and Systems Security*, vol. 1, no. 2, pp. 114–136, Jun 2017. [Online]. Available: https://doi.org/10.1007/s41635-017-0013-2 33

[77] A. Cañete, M. Amor, and L. Fuentes, "Supporting the evolution of applications deployed on edge-based infrastructures using multi-layer feature models," in *Proceedings of the 24th ACM International Systems and Software Product Line Conference - Volume B*, ser. SPLC '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 79–87. [Online]. Available: https://doi.org/10.1145/3382026.3425772 35

[78] A. Cañete, K. Djemame, M. Amor, L. Fuentes, and A. Aljulayfi, "A proactive energy-aware auto-scaling solution for edge-based infrastructures," in *2022 IEEE/ACM 15th International Conference on Utility and Cloud Computing (UCC)*, 2022, pp. 240–247. 37, 74

[79] R. T. Geraldi, S. Reinehr, and A. Malucelli", "Software product line applied to the Internet of Things: A systematic literature review," *Information and Software Technology*, vol. 124, p. 106293, 2020. 41

[80] D. Dhungana, P. Grünbacher, R. Rabiser, and T. Neumayer, "Structuring the modeling space and supporting evolution in software product line engineering," *Journal of Systems and Software*, vol. 83, no. 7, pp. 1108 – 1122, 2010, sPLC 2008. 42

[81] M.-O. Reiser and M. Weber, "Multi-level feature trees," *Requirements Engineering*, vol. 12, no. 2, pp. 57–75, Apr 2007. 42

[82] G. Holl, P. Grünbacher, and R. Rabiser, "A systematic review and an expert survey on capabilities supporting multi product lines," *Information and Software Technology*, vol. 54, no. 8, pp. 828 – 852, 2012. 42

[83] M. Acher, P. Collet, A. Gaignard, P. Lahire, J. Montagnat, and R. B. France, "Composing multiple variability artifacts to assemble coherent workflows," *Software Quality Journal*, vol. 20, no. 3, pp. 689–734, 2012. 42

[84] M. Acher, P. Collet, P. Lahire, and R. B. France, "Familiar: A domain-specific language for large scale management of feature models," *Science of Computer Programming*, vol. 78, no. 6, pp. 657 – 681, 2013. 42

[85] E. Farahani and J. Habibi, "Feature model configuration based on two-layer modelling in software product lines," *International Journal of Electrical and Computer Engineering*, vol. 9, pp. 1–11, 03 2019. 42

[86] D. Rabiser, H. Prähofer, P. Grünbacher, M. Petruzelka, K. Eder, F. Angerer, M. Kromoser, and A. Grimmer, "Multi-purpose, multi-level feature modeling of large-scale industrial software systems," *Software & Systems Modeling*, vol. 17, 10 2016. 42

[87] A. Abbas, I. Farah Siddiqui, S. U. Lee, A. Kashif Bashir, W. Ejaz, and N. M. F. Qureshi, "Multi-objective optimum solutions for iot-based feature models of software product line," *IEEE Access*, vol. 6, pp. 12 228–12 239, 2018. 42

[88] J. Guo, J. White, G. Wang, J. Li, and Y. Wang, "A genetic algorithm for optimized feature selection with resource constraints in software product lines," *Journal of Systems and Software*, vol. 84, no. 12, pp. 2208 – 2221, 2011. 42

[89] T. Zhao, S. Zhou, X. Guo, Y. Zhao, and Z. Niu, "A cooperative scheduling scheme of local cloud and internet cloud for delay-aware mobile cloud computing," in *IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2015, pp. 1–6. 43

[90] Y. Ge, Y. Zhang, Q. Qiu, and Y. Lu, "A game theoretic resource allocation for overall energy minimization in mobile cloud computing system," in *ISLPED'12-Proceedings of the International Symposium on Low Power Electronics and Design*, Sep. 2012, pp. 279–284. 43

[91] M. S. Elbamby, M. Bennis, and W. Saad, "Proactive edge computing in latency-constrained fog networks," in *2017 European Conference on Networks and Communications (EuCNC)*, 2017, pp. 1–6. 43

[92] R. Urgaonkar, S. Wang, T. He, M. Zafer, K. Chan, and K. K. Leung, "Dynamic service migration and workload scheduling in edge-clouds," *Performance Evaluation*, vol. 91, pp. 205–228, 2015, special Issue: Performance 2015. 43

[93] M. Chen, M. Dong, and B. Liang, "Joint offloading decision and resource allocation for mobile cloud with computing access point," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2016, pp. 3516–3520. 43

[94] Y. Al-Dhuraibi, F. Paraiso, N. Djarallah, and P. Merle, "Elasticity in cloud computing: State of the art and research challenges," *IEEE Transactions on Services Computing*, vol. 11, no. 2, pp. 430–447, 2018. 44

[95] A. M. Medhat, T. Taleb, A. Elmangoush, G. A. Carella, S. Covaci, and T. Magedanz, "Service function chaining in next generation networks: State of the art and research challenges," *IEEE Communications Magazine*, vol. 55, no. 2, pp. 216–223, 2017. 46

[96] K. Kaur, V. Mangat, and K. Kumar, "A comprehensive survey of service function chain provisioning approaches in sdn and nfv architecture," *Computer Science Review*, vol. 38, p. 100298, 2020. 46

[97] M. Aazam, S. Zeadally, and K. A. Harras, "Offloading in fog computing for IoT: Review, enabling technologies, and research opportunities," *Future Generation Computer Systems*, vol. 87, pp. 278 – 289, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X18301973 46

[98] D. Li, P. Hong, K. Xue, and J. Pei, "Virtual network function placement and resource optimization in nfv and edge computing enabled networks," *Computer Networks*, vol. 152, pp. 12–24, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128618305000 46

[99] A. Leivadeas, M. Falkner, I. Lambadaris, and G. Kesidis, "Optimal virtualized network function allocation for an sdn enabled cloud," *Computer Standards & Interfaces*, vol. 54, pp. 266–278, 2017, sI: Standardization SDN—&NFV. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0920548917300168 46, 47

[100] M. Huang, W. Liu, T. Wang, A. Liu, and S. Zhang, "A cloud–mec collaborative task offloading scheme with service orchestration," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 5792–5805, 2020. 46, 47

[101] Z. Yang, R. Gu, H. Li, and Y. Ji, "Approximately lossless model compression-based multilayer virtual network embedding for edge-cloud collaborative services," *IEEE Internet of Things Journal*, pp. 1–1, 2023. 46

[102] Y. Yue, B. Cheng, X. Liu, M. Wang, B. Li, and J. Chen, "Resource optimization and delay guarantee virtual network function placement for mapping sfc requests in cloud networks," *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1508–1523, 2021. 46

[103] M. Bunyakitanon, A. P. da Silva, X. Vasilakos, R. Nejabati, and D. Simeonidou, "Auto-3p: An autonomous vnf performance prediction & placement framework based on machine learning," *Computer Networks*, vol. 181, p. 107433, 2020. 46, 47

[104] T. Subramanya, D. Harutyunyan, and R. Riggio, "Machine learning-driven service function chain placement and scaling in mec-enabled 5g networks," *Computer Networks*, vol. 166, p. 106980, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128619310254 46, 47

[105] Z. Yang, R. Gu, and Y. Ji, "Virtual network function placement based on differentiated weight graph convolutional neural network and maximal weight matching," in *2021 IEEE Symposium on Computers and Communications (ISCC)*, 2021, pp. 1–7. 46

[106] Y. Cheng, L. Yang, and H. Zhu, "Deployment of service function chain for nfv-enabled network with delay constraint," in *2018 International Conference on Electronics Technology (ICET)*, 2018, pp. 383–386. 46

[107] O. Soualah, M. Mechtri, C. Ghribi, and D. Zeghlache, "Energy efficient algorithm for vnf placement and chaining," in *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, 2017, pp. 579–588. 46, 47

[108] C. Assi, S. Ayoubi, N. El Khoury, and L. Qu, "Energy-aware mapping and scheduling of network flows with deadlines on vnfs," *IEEE Transactions on Green Communications and Networking*, vol. 3, no. 1, pp. 192–204, 2019. 46, 47

[109] N. H. Thanh, N. Trung Kien, N. V. Hoa, T. T. Huong, F. Wamser, and T. Hossfeld, "Energy-aware service function chain embedding in edge–cloud environments for iot applications," *IEEE Internet of Things Journal*, vol. 8, no. 17, pp. 13 465–13 486, 2021. 46, 47

[110] R. Moosavi, S. Parsaeefard, M. A. Maddah-Ali, V. Shah-Mansouri, B. H. Khalaj, and M. Bennis, "Energy efficiency through joint routing and function placement in different modes of sdn/nfv networks," *Computer Networks*, vol. 200, p. 108492, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128621004345 46, 47

[111] G. Sun, Y. Li, H. Yu, A. V. Vasilakos, X. Du, and M. Guizani, "Energy-efficient and traffic-aware service function chaining orchestration in multi-domain networks," *Future Generation Computer Systems*, vol. 91, pp. 347–360, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X1831848X 46, 47

[112] G. MIRJALILY and Z. LUO, "Optimal network function virtualization and service function chaining: A survey," *Chinese Journal of Electronics*, vol. 27, pp. 704–717, 07 2018. 47

[113] S. K. Datta, C. Bonnet, and N. Nikaein, "Self-adaptive battery and context aware mobile application development," in *2014 International Wireless*

Communications and Mobile Computing Conference (IWCMC), Aug 2014, pp. 761–766. 48

[114] A. A. Nacci, M. Mazzucchelli, M. Maggio, A. Bonetto, D. Sciuto, and M. D. Santambrogio, "morphone.OS: context-awareness in everyday life," in *Digital System Design (DSD)*, 2013, pp. 779–786. 48

[115] N. Z. Naqvi, J. Devlieghere, D. Preuveneers, and Y. Berbers, "Mascot: Self-adaptive opportunistic offloading for cloud-enabled smart mobile applications with probabilistic graphical models at runtime," in *49th Hawaii International Conference on System Sciences (HICSS)*, 2016, pp. 5701–5710. 48

[116] P. A. d. S. Duarte, F. M. Barreto, F. A. d. A. Gomes, W. V. d. Carvalho, and F. A. M. Trinta, "CRITiCAL: A configuration tool for context aware and mobile applications," in *Computer Software and Applications Conference*, vol. 2, 2015, pp. 159–168. 48

[117] J. Floch, C. Frà, R. Fricke, K. Geihs, M. Wagner, J. Lorenzo, E. Soladana, S. Mehlhase, N. Paspallis, H. Rahnama, P. Ruiz, and U. Scholz, "Playing MUSIC — building context-aware and self-adaptive mobile applications," *Software: Practice and Experience*, vol. 43, no. 3, pp. 359–388, 2013. 48

[118] G. G. Pascual, M. Pinto, and L. Fuentes, "Self-adaptation of mobile systems driven by the common variability language," *Future Generation Computer Systems*, vol. 47, pp. 127–144, 2015. 48

[119] M. Linares-Vásquez, C. Bernal-Cárdenas, G. Bavota, R. Oliveto, M. Di Penta, and D. Poshyvanyk, "Gemma: Multi-objective optimization of energy consumption of guis in android apps," in *International Conference on Software Engineering Companion (ICSE-C)*, May 2017, pp. 11–14. 48

[120] A. Banerjee and A. Roychoudhury, "Automated re-factoring of android apps to enhance energy-efficiency," in *International Conference on Mobile Software Engineering and Systems (MOBILESoft)*, May 2016, pp. 139–150. 48

[121] R. Morales, R. Saborido, F. Khomh, F. Chicano, and G. Antoniol, "Earmo: An energy-aware refactoring approach for mobile apps," *IEEE Transactions on Software Engineering*, vol. 44, no. 12, pp. 1176–1206, Dec 2018. 48