Computer Science and Information Systems 23(1):231–253 https://doi.org/10.2298/CSIS250915076B

# Meeting Consumer Demand: Supporting Efficient Services Deployment for Cloud-Connected Devices

Iván Bernabé-Sánchez[1], Daniel Díaz-Sánchez[2], Alberto Fernández[1], Holger Billhardt[1], and Sascha Ossowski[1]

[1] CETINIA, Universidad Rey Juan Carlos
Tulipán s/n, 28933 Móstoles (Madrid)
{ivan.bernabe,alberto.fernandez,holger.billhardt,sascha.ossowski}@urjc.es
[2] Universidad Carlos III de Madrid
Avda. de la Universidad 30, 28911 Leganés (Madrid)
dds@it.uc3m.es

**Abstract.** The demand for digital services driven by the growth of internet-connected devices and consumer needs is constantly increasing. Therefore, the provision of services to achieve a good user experience is becoming crucial. Network Functions Virtualization (NFV) has been established as a key technology for service providers. NFV decouples network functions from hardware devices by allowing network services, called virtualized network functions (VNF), to be hosted on commodity hardware, simplifying and making service deployment more flexible. The placement of VNFs within infrastructure presents a critical technical challenge due to its direct impact on network performance, reliability, and operational costs. This article presents a service architecture that can be integrated with network service managers to identify priorities that should be taken into account when deploying VNFs or services in infrastructures. The proposed solution collects information about the network service to be deployed and analyzes and evaluates its requirements by comparing them with operating patterns. As a result, recommendations for the deployment of each service are obtained. The solution also proposes a provisioning service which, based on the generated recommendations, adapts the configuration files for deployment and connects with the configuration management tools to automate the deployment.

**Keywords:** VNF placement, Resource Allocation, Virtualization, Cloud Computing, Edge Computing, Network function virtualization, Service function chain.

## 1. Introduction

The proliferation of end-user devices and the emergence of new mobile technologies, such as smartphones, wearables, and IoT sensors—have sharply increased mobile data consumption [8]. This trend is driven by enhanced device capabilities (e.g., high-resolution displays and HD cameras) and the growing demand for immediate access to digital content. As a result, telecommunication service providers (TSPs) are under increasing pressure to adapt their infrastructures, facing challenges related to resource optimization, hardware rigidity, and inflexible control mechanisms. These factors increase capital and operating expenditures, negatively impacting profitability [35].

Despite advances in cloud computing and Network Functions Virtualization (NFV), several challenges remain. Efficient resource allocation, optimal configuration selection,

and effective distribution of virtualized network functions (VNFs) remain complex problems, particularly in heterogeneous and distributed environments that include edge computing [19, 9]. Existing solutions often optimize individual metrics, such as latency or resource use, overlooking holistic strategies that address the full complexity of these infrastructures.

To address these issues, this work introduces an architecture that automates the deployment and configuration of VNFs. The system leverages optimization techniques that consider geographic location, infrastructure compatibility, and service-specific requirements to recommend suitable deployment configurations.

The proposed approach addresses the following key challenges:

- *Optimal Configuration Selection for VNFs*: As Service Function Chains (SFCs) grow in complexity, deployment strategies must optimize resources, minimize latency, and ensure component compatibility across infrastructures.
- *Management of Heterogeneous Infrastructures*: Cloud-edge integration introduces disparities in capacity, location, and technology, complicating orchestration.
- *Automation of Resource Allocation*: Real-time resource provisioning is essential to meet fluctuating demands while preserving Quality of Service (QoS).
- *Validation in Real-World Scenarios*: Evaluation under realistic conditions such as multimedia streaming, ensures broader applicability and reliability.

This article makes the following key contributions:

- *Architecture for Efficient VNF Deployment*: A systematic approach is proposed for the automated selection and configuration of VNFs in distributed environments.
- *Optimization of Distributed Service Management*: The solution integrates edge and cloud resources to reduce latency and enhance user experience in latency-sensitive applications.
- *Automated Configuration Selection System*: A decision model evaluates infrastructure compatibility, service demands, and optimal placement to guide deployment.
- *Simulation-Based Validation*: A multimedia streaming scenario demonstrates the effectiveness of the system, using latency and performance metrics as benchmarks.

The remainder of this article is structured as follows. Section 2 discusses related work; Section 3 presents the proposed architecture; Section 4 explains configuration selection mechanisms; Section 5 details the implementation and results and Section 6 provides conclusions and outlines future research directions.

## 2. Related work

A multitude of papers can be found in the literature proposing different strategies to decide the location of VNFs. This decision process takes into account relevant factors such as the characteristics and resources of the available nodes [20], the capacities of the VNF nodes to be deployed [31], the traffic and capacity of the network in which they are to be deployed, the energy consumption and management of the systems [27, 25] or even the use of data centers located in the cloud or at the edge of the network [24]. All these factors condition the techniques and strategies for the optimization of VNF deployment systems, which has generated different solutions to address the problem. For example, works such

as [28] and [4] focus on optimizing latency when VNFs are deployed. Other works such as [34], [1] and [37] take into account network traffic to decide the placement of VNFs. In [16] customer satisfaction is taken into account as the main objective to decide where to deploy VNFs. In [4] and [11] the authors also consider the costs associated with the deployment of VNFs.

Other works are oriented towards managing the deployment of services on top of compute nodes located at the fog layer and taking into account requirements related to deployment cost, required computation, or energy consumption. These works consider services that can perform all kinds of tasks, from virtualised network functions (VNF) to other types of tasks related to various domains.

In [29], a solution for resource management is presented in the fog layer that integrates fog computing, machine learning, and context-aware computing to provide efficient resource management. [29] takes into account compute, buffer and disk space requirements. In [13], a method is proposed for selecting edge nodes in the smart manufacturing domain to minimize the total cost of edge node deployment, network delay, packet loss, and energy consumption. To this end, they proposed an algorithm based on game theory to select the most appropriate nodes to deploy services.

In [23] a recommender framework is proposed to decide whether to run services on nodes located on the edge or in a cloud datacenter.

Finding suitable positions to place VNFs or services in a way that satisfies the requirements is a critical problem. It has been shown that this problem is NP-hard [3] so, due to its difficulty, finding the optimal solution for VNF Placement (VNF-P), especially in large-scale network scenarios, is impossible. To address the problem of VNF placement in small- to moderate-scale environments, VNF-P can be formulated as Integer Linear Programming (ILP) and Mixed ILP (MILP) problems. [2] and [15] use ILP and MILP to calculate the optimal placement. In [2], ILP is used with the objective of finding the optimal placement of an Evolved Packet Core (EPC) service considering latency limits in a mobile network scenario. In [15] the authors also study the VNF placement problem for big data processing by providing a MILP model where they use a simple relaxation-based heuristic algorithm to minimize the overall communication cost. However, ILP models are unsuitable for large systems and only apply to scenarios where all variables are integers.

In order to help reduce the energy consumption of consumer electronics, [36] presents a solution to address Mixed Integer NonLinear Programming (MINLP) problems of bandwidth and computational resource allocation by carefully considering the trade-off between computational complexity and implementation efficiency.

Most of the works take the optimization of costs, energy, network traffic, available resources in the nodes, and latency as the main factors. However, we consider that it is also important to decide on other relevant aspects, even before launching the planning processes for the deployment of VNFs in the works discussed. For example, some solutions will provide us with the ideal location for a VNF based on network traffic or energy expenditure. However, it is important to assess the service requirements that will directly influence its operation. Once the service requirements have been identified, we propose to launch the optimization process for the deployment of VNFs. However, unlike these approaches, our proposal does not focus on optimizing the runtime performance of services or applications. Instead, we emphasize a preliminary but essential phase: identifying the service requirements that will directly influence their operation before initiat-

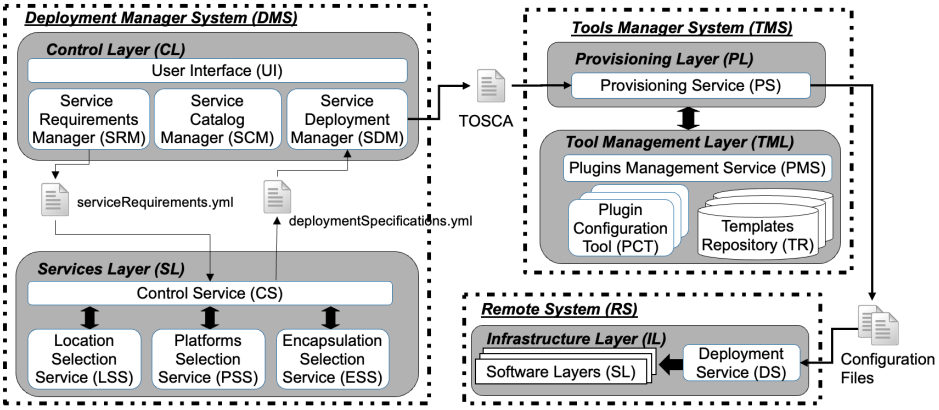



**Fig. 1.** Main blocks of the architecture presented in this article and their relationships. The proposed solution allows to manage the deployment of services on a service architecture

ing any deployment planning or optimization process. By assessing these requirements in advance, our approach focuses on generating optimal deployment configurations that enhance compatibility with heterogeneous infrastructures, simplify management, and improve the overall deployment process of applications and services. This perspective introduces a complementary dimension to the state of the art, addressing aspects generally assumed or overlooked in existing VNF and service placement solutions.

## 3. Solution architecture

To achieve the objectives outlined in Section 1, we have designed a novel architecture capable of providing the necessary functionalities. Although existing architectures facilitate service deployments in distributed cloud and edge environments, they often lack comprehensive support for the selection and optimization of deployment configurations, which is a key focus of our proposed solution.

Solutions such as T-NOVA[21], SONATA NFV[7], OSM [30], ONAP (Open Network Automation Platform)[6], Cloudify[10], and the MEC (Multi-access Edge Computing) Framework [12] offer valuable functionalities for infrastructure resource management, cloud and edge service deployment, and task automation. Each also provides specialized capabilities tailored to specific domains or needs. However, the architecture designed in this article differs mainly from these solutions in that it can consider specific factors such as infrastructure compatibility, location selection, and configuration of the service according to its needs and characteristics, taking them into account when deploying the service.

Fig. 1 illustrates the main blocks of our proposed architecture and their interrelationships. This solution enables the automated deployment and configuration of service-based systems. The architecture is divided into three main blocks: i) Deployment Manager System (DMS); ii) Tools Manager System (TMS); iii) Remote System (RS). Each block is described below.

### 3.1. Deployment Manager System (DMS)

The DMS offers tools to recommend the configuration of services and infrastructures before those services are deployed. Service providers (SPs) use the DMS to obtain configurations adapted to the operational requirements of the service to be deployed. When SPs want to deploy a new service, they could use the solution proposed in this work to obtain the configuration of the software layers where the services are to be deployed and then deploy them.

SPs interact with the DMS through the control layer (CL) which has a user interface (UI) that allows them to enter the description and requirements of the services. The Service Requirements Manager (SRM) is responsible for obtaining information on the operating requirements of the service to be deployed. As a result, the SRM writes service specifications and provides information in an understandable format for the modules contained in the SL.

The CL also integrates the Services Catalog Manager and Service Deployment Manager. The Service Catalogue Manager (SCM) maintains the information of services that have been previously registered. Facilitates the deployment of services that have been previously used, as it has a registry of previous deployment configurations. The Service Deployment Manager (SDM) receives the deployed specifications generated by the SL. The user can see the recommended options to deploy the desired service. In addition, the SDM provides the deployment specifications for the Tools Manager System (TMS). The TMS transforms the deployment specifications into configuration files that can be understood by configuration management tools.

The service layer (SL) is responsible for processing the information received from the CL. Upon receiving service requirements from the SRM, the SL initiates a process to determine the most suitable deployment configurations by evaluating three key aspects: i) the location of the service to be deployed; ii) the software architecture on which it will be deployed; and iii) the encapsulation technology. Each of these aspects is processed separately.

The SL uses the Location Selection Service (LSS) to decide whether the service should be deployed in an infrastructure located at the edge of the network or in a data center hosted in the cloud. Then, the SL uses the Platform Selection Service (PSS) to decide whether the service should be installed on a server or cloud architecture. Finally, the Encapsulation Selection Service (ESS) is responsible for evaluating the available options in terms of encapsulation technology. The ESS will decide whether it is more convenient to encapsulate a service inside a heavy or light virtualization container or even decide that it is better not to use virtualization.

### 3.2. Tools Manager System (TMS)

The TMS is in charge of deploying services over remote infrastructures by making use of different configuration management tools [17]. The TMS receives the service specifications and transforms them into an understandable format for configuration management tools (CMT). CMTs are responsible for facilitating and automating the configuration management of computer systems, applications and other software components [18]. In addition, CMTs favor maintenance update management [14], therefore they play an important role in software maintenance.

The TMS is composed of two layers: the Provisioning Layer (PL) and the Tool Management Layer (TML). The PL receives from the CL the service to be deployed and recommendations on how the service should be deployed (e.g., where the service should be deployed and whether or not virtualization technology should be used). For instance, if the software requires high processing capacity with low latencies, it might be better to run it on a system that does not have a virtualization layer between the host operating system and the service to be deployed, so that latency can be avoided. In such a case, the TMS receives from the DMS the specifications where the latency requirement is contemplated and proposes a latency-friendly deployment configuration. The Provisioning Service (PS) is able to process the recommendations proposed by the DMS with the objective of translating them into a format that is understandable by the configuration management tool installed in the service provider infrastructure. To perform this adaptation, the PL connects to the Plugins Management Service (PMS), which provides templates and configuration parameters specific to the selected configuration tool. Here, we propose a plugin-based system where each plugin contains relevant information about a configuration tool in order to enable the system to work with different such tools. This plugin-based design allows us to have a catalogue of configuration tools available in the system. In addition, plugins can be easily added, removed and even updated without affecting the operation of the rest of the system. The PMS consults the Plugin Configuration Tool and obtains information depending on the selected tool. In addition, it provides parameters and configuration templates to the PS which are necessary to make configuration files understandable to the configuration tool selected by the SP. Configuration management tools use local agents to facilitate management tasks on remote systems. The agents are deployed on the remote nodes and are responsible for implementing the instructions and commands received from the central node. Finally, the PS generates a package of configuration files according to the specifications received from the PMS.

### 3.3. Remote System (RS)

The RS represents the infrastructure on which the desired service will be installed. The RS is composed of an infrastructure layer where software services are deployed. This infrastructure layer can contain multiple software layers, each providing specific capabilities or technologies. Examples of software layers are the Java Virtual Machine or Microsoft's .NET framework, both providing environments for running applications.

### 3.4. DMS, TMS and RS Working Together

Our proposed solution integrates a set of distinct services, functionally grouped into the DMS, TMS, and RS, serving as a central point for various participants within a digital service ecosystem. Each participant in this ecosystem assumes a specific role based on their potential actions (see Fig. 2). The process unfolds as follows:

1. Service request by the consumer. A consumer requests a service from a Service Provider (SP).
2. Checking service availability. The SP queries the service catalogue to determine whether the requested service is already deployed in the available infrastructures.

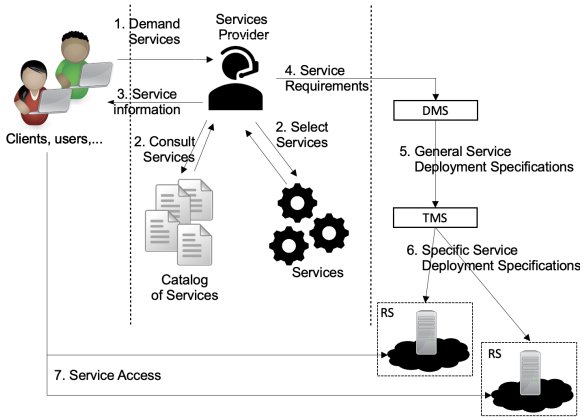



**Fig. 2.** Types of users, actions and flow of information exchanged between the main blocks of the proposed system and the users

3. Providing access when the service is already deployed. If the service exists and meets the operational requirements, the SP provides the consumer with the necessary information to access it.
4. Forwarding requirements when the service is not deployed. If the service is not deployed, the SP forwards the service requirements to the DMS.
5. Generating deployment specifications. The DMS processes the received requirements and generates a deployment specification in a TOSCA format file [22].
6. Adapting the deployment to the target infrastructure. Using the TOSCA file, the SP interacts with the TMS to generate the final deployment specifications tailored to the characteristics of the selected Remote System (RS).
7. Service deployment and access. Once the services have been deployed on the RS, consumers can access the requested service

## 4. Configuration Selection Process

As seen in the previous section, the DMS is able to analyze service requirements and decide which technological execution environment is the most suitable for the services to be deployed. This section shows the mechanisms used to select the most appropriate deployment configuration. For this purpose, it is necessary to define the requirements of the services considered in the selection process. Each requirement is defined by one or more qualitative values (Low, Medium, High). For example, a service may require a low boot time ({Low}), while another may accept low or medium ({Low, Medium}). This simple and well-defined way of describing the requirements, rather than using numerical values, facilitates their interpretation and specification by service providers. The list of identified service requirements $SR = \{r_1, r_2, ..., r_{16}\}$ is presented below:

– Time to be transferred ($r_1$). Time taken to transfer a given service to deploy it in another location or system.

- Time To Automatically Deploy ($r_2$). Time taken to deploy a given service.
- Boot time ($r_3$). Time required for a service to start operation, i.e. to transition from the stopped state to the running state.
- Compatibility ($r_4$). Ability of the service to be seamlessly deployed across a variety of infrastructures.
- Easy of Portability ($r_5$). Complexity of a service when it is deployed on different infrastructures.
- Isolation Degree ($r_6$). Degree of preference for a service to be encapsulated. Encapsulating a service facilitates its mobility (because it contains the configuration of the service) and provides security to it (deploying a service in a virtualised environment provides an additional software layer).
- Consumption of Resources ($r_7$). Indicates whether a service normally requires few, many, or an intermediate level of computational resources.
- Easy to manage ($r_8$). Degree of complexity of a service to be deployed.
- Scalability ($r_9$). The ability to dynamically allocate computational resources to a service that is running.
- Hardware Fault Tolerant ($r_{10}$). Indicates whether a service requires the infrastructure to be fault-tolerant at the hardware level.
- Cost ($r_{11}$). The cost of the use of the infrastructure where the service is deployed.
- Latency ($r_{12}$). Is the latency required on network connections.
- Jitter ($r_{13}$). The jitter of the network connection that the service is used.
- Distance to User ($r_{14}$). The physical distance between the deployed service and the users consuming it.
- Location Awareness ($r_{15}$). Used to indicate whether the service is strongly bound to a specific location or can be used in other available locations.
- Centralization Degree ($r_{16}$). Indicates whether the service should be deployed in a centralized location or not.

The deployment configuration selection process takes into account the service requirements and considers three dimensions of the execution environment, namely (i) the encapsulation technology, (ii) the deployment infrastructure, and (iii) the location. They are detailed in the following subsections.

### 4.1. Encapsulation technology selection

During the analysis carried out in this work, the following three types of encapsulation have been identified, each of them with different characteristics:

- *Heavy virtualization (HV)*: It is characterized by the installation of services in an operating system encapsulated in a virtual machine. The HV emulates the operating system and the hardware resources.
- *Light virtualization (LV)*: Similar to heavy virtualization based on virtual machines, but in this case, no hardware resources are virtualised and there is no need to install a new operating system.
- *No virtualization (NV)*: It is characterized by the installation of the service directly on the operating system and the service is not encapsulated.

**Table 1.** Comparison of deployment characteristics depending on the virtualization technology used. No virtualization (NV), light virtualization (LV) and heavy virtualization (HV)

| Characteristic | NV | HV | LV |
|---|---|---|---|
| T. to be transferred (c1) | High | Medium | Low |
| T. to automatic deployment (c2) | High | Medium | Low |
| Boot time (c3) | High | Medium | Low |
| Compatibility (c4) | High | Medium | Low |
| Easy portability (c5) | Low | Medium | High |
| Isolation degree (c6) | High | Medium | Low |
| Consumption of resources (c7) | Low | Medium | Low |
| Easy to manage (c8) | High | Medium | Low |

Normally, network functions are deployed in virtualized environments based on virtual machines [2, 26]. However, the use of VMs is not always the best option due to performance requirements which sometimes cannot be met by this technology (e.g. low start-up or high transfer times). For this reason, telecommunication service providers (TSPs) have explored other alternatives such as container network functions (CNFs) [5]. CNFs consist of running network services in containers instead of VMs, providing network functions (NFs) with performance similar to that they would have if they were running in a non-virtualized environment.

Containers are based on virtualization [32] and run as an application on top of a general-purpose operating system. This approach allows to boost efficiency through the elimination of the overhead produced by the virtualization of resources and drivers for the management of hardware resources in VMs. In addition, containers reduce system overhead compared to VMs. This model can run a multitude of isolated containers that share the operating system kernel. The CS collects the requirements for the services to be evaluated and then applies a selection process to obtain a numerical result for each of the types of encapsulation. The type of encapsulation in which a service is deployed has an impact on its performance, providing unique and different capabilities compared to other types of encapsulation.

Table 1 presents a comparison of how different encapsulation technologies impact key service characteristics ($c_i$) relevant to their selection.

For the encapsulation type selection process, our proposed mechanism compares a specific subset of service requirements ($SR_E = \{r_1, r_2, r_3, r_4, r_5, r_6, r_7, r_8\}$) with the characteristics of each encapsulation type presented in Table 1.

Let $E = \{NV, HV, LV\}$ be the set of encapsulation types. The Encapsulation Selection Service (ESS) uses expression (1) to obtain a numerical value ($OP_e$) that indicates the number of requirements that are fulfilled for the service being evaluated.

$$OP_e = \sum_{r \in SR_E} ET_{re}, \forall e \in E \tag{1}$$

$$ET_{ie} = \begin{cases} 1, & \text{if } v_{ie} \in r_i \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

**Table 2.** Example of obtaining the value of $OP_E$ as the number of matches between the service requirements ($SR_E$) and the characteristics of each encapsulation technology

| | $SR_E$ | $NV$ | $HV$ | $LV$ |
|---|---|---|---|---|
| c1 | Low | High | Medium | Low |
| c2 | Low | High | Medium | Low |
| c3 | Low | High | Medium | Low |
| c4 | {Low, Medium} | High | Medium | Low |
| c5 | {Medium, High} | Low | Medium | High |
| c6 | {Low, Medium, High} | High | Medium | Low |
| c7 | {High} | Low | Medium | Low |
| c8 | {High} | High | Medium | Low |
| $OP_E$ | | 2 | 3 | 6 |

where $v_{ie}$ is the value of characteristic $c_i$ for the type of encapsulation $e$. Finally, the decision on the recommended encapsulation type ($V_E$) is made according to the expression (3).

$$V_E = \arg\max_{e \in E} OP_e \tag{3}$$

Table 2 shows an example. The last row ($OP_E$) shows the results of applying equation (1) to each column $NV$, $HV$ and $LV$ of Table 2. In this example, the environment $LV$ is the most appropriate to deploy that service.

### 4.2. Deployment infrastructure selection

This section shows the Platform Selection Service (PSS) mechanisms to obtain the optimal infrastructure when a given service is to be deployed. In this work, we have identified two possible infrastructures to consider:

- *Infrastructure based on virtual private servers (VPS)*: This infrastructure is characterized by having all the physical resources available for the services.
- *Cloud-based infrastructure (CC)*: This infrastructure is characterized by having a layer that manages the infrastructure resources transparently to the upper layers. The execution instances are managed by a hypervisor that manages the infrastructure resources globally and allocates them to the instances depending on their needs. The services and applications are deployed and executed in those instances.

VPS is a fixed block of physical server resources, while cloud-based infrastructures are based on multiple dedicated servers. VPS provides better customization options, resource allocation and speed and is more affordable than cloud-based infrastructures. In addition, VPS allows for near-instant scalability and also avoids noisy neighbor syndrome. This syndrome occurs in situations where a virtual machine (VM) monopolizes resources (such as bandwidth, disk I/O, and CPU) in a shared infrastructure. This causes performance issues for other virtual machines that share the same physical resources, which can lead to uneven network performance, increased CPU timeouts, disk I/O accesses, and more. In the case of cloud-based infrastructures, they offer higher uptime than VPS hosting due to their redundant system. Cloud hosting ensures high availability due to multiple servers in

**Table 3.** Comparison of the characteristics depending on the type of infrastructures used

| | VPS | Cloud computing |
|---|---|---|
| Scalability (c9) | Low | High |
| Hardware fault tolerant (c10) | Low | High |
| Cost (c11) | Low | High |

a redundant system. If one server fails, files and applications are immediately migrated to another server without stopping applications and services from running. Memory capacity and CPU power are also expanded on demand to meet the needs of another customer or compensate for those without affecting other users. The CS collects the requirements registered in the Platform Selection Service (PSS) related to the service to be evaluated. Choosing the right infrastructure for a service is important because it influences the behavior of the service, and therefore the user's experience when consuming that service. Table 3 presents a comparison of key service characteristics relevant to the selection of the deployment infrastructure. The selection process works similarly to the one in Section 4.1. Table 3 collects the characteristics related to different types of deployment infrastructures. The PSS, similar to the ESS, also models the service requirements through $SR_I = \{r_9, r_{10}, r_{11}\}$. Then, the selection process assigns a numerical value to each deployment type presented in the columns of Table 3. Let $I = \{VPS, CC\}$ be the set of deployment types. The Platform Selection Service (PSS) uses the expression (4) to obtain a numerical value ($OP_i$) that indicates the number of requirements that are fulfilled for the service being evaluated.

$$OP_i = \sum_{r \in SR_I} IT_{ri}, \forall i \in I \tag{4}$$

$$IT_{ji} = \begin{cases} 1, & \text{if } v_{ji} \in r_j \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

where $v_{ji}$ is the characteristic value $c_j$ for infrastructure type $i$. Finally, the decision on the recommended infrastructure type ($V_I$) is made according to the expression (6).

$$V_I = \arg\max_{i \in I} OP_i \tag{6}$$

### 4.3. Location selection

This section discusses the decision process for deploying a service in a centralized or a decentralized data center. This is an important aspect when a service is to be deployed because it can influence the quality of the services. Edge computing and cloud computing [33] are two different technologies that serve different goals. Edge computing and cloud computing are different technologies that cannot replace each other. However, they can work together to provide an end-to-end solution. Cloud computing is more suitable for applications where large amounts of data need to be intensively processed at once (e.g., image recognition), can handle large-scale workloads, and can also provide centralized storage in large-scale data centers allowing access from anywhere with an Internet connection. On the other hand, edge computing is best suited for its use in applications that

**Table 4.** Comparison of the characteristics depending on each option of localization: Centralized Computing (CC), Edge Computing (EC)

| | CC | EC |
|---|---|---|
| Latency (c12) | High | Low |
| Jitter (c13) | High | Low |
| Distance to the user (c14) | High | Low |
| Location awareness (c15) | Low | High |
| Centralization degree (c16) | High | Low |

require real-time data processing and inter-device communication, processing real-time sensitive data, using in remote locations with limited or no network connectivity, providing local storage, etc.

This work takes into account the requirements and needs that are part of the selection process when the location of the infrastructure to deploy a service has to be decided. In this work, two locations have been identified and are listed in Table 4.

The Location Selection Service (LSS) applies a similar selection process, taking into account a specific subset of the service requirements relevant to the location of deployment $SR_L = \{r_{12}, r_{13}, r_{14}, r_{15}, r_{16}\}$ and compares them with the characteristics shown in Table 4.

We define $L = \{CC, EC\}$ as the set of localization types, where $CC$ represents centralized computing in a specific location and $EC$ represents edge computing distributed in different localizations.

The LSS uses expression (7) to obtain a numerical value that indicates the number of requirements that are fulfilled for the service being evaluated.

$$OP_l = \sum_{r \in SR_L} LT_{rl}, \forall l \in L \tag{7}$$

$$LT_{il} = \begin{cases} 1, & \text{if } v_{il} \in r_i \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

where $v_{il}$ is the value of characteristic $c_i$ for the type of location $l$. Finally, the decision on the recommended location ($V_L$) is made according to the expression (9).

$$V_L = \arg\max_{l \in L} OP_l \tag{9}$$

## 5. Implementation and results

This section shows a prototype and the results of the tests performed on the proposed solution. To evaluate the work presented in this article, the deployment of a video streaming service has been considered. This service provides multimedia content to clients who connect to consume such content. Listing 1.1 presents the content of the *serviceRequirements.yml* file, which defines the requirements for the video streaming service used in our evaluation.

**Listing 1.1.** Code contained in the file serviceRequirements.yml

```
1  Service:
2    Name: VideoService
3    Description: It is a video server
4    timeToBeTransfered: low
5    timeToAutomaticallyDeploy: low
6    bootTime: low
7    compatibility: [low, medium]
8    easyOfPortability: [medium, high]
9    isolationDegree: [low, medium, high]
10   consumptionOfResources: high
11   easyToManage: high
12   scalability: high
13   hardwareFaultTolerant: low
14   cost: medium
15   latency: low
16   jitter: low
17   distanceToUser: medium
18   locationAwareness: low
19   centralizationDegree: medium
```

**Listing 1.2.** Code contained in the file *deploymentSpecification.yml*, which contains the results obtained after applying the selection process by LSS, PSS and ESS

```
1  V_E: LV
2  V_I: VPS
3  V_L: EC
```

Lines 2-3 are used to assign a name and description to the service, and the other lines are used to define the requirements of the service, which have been described in Section 4.

Specifically, the requirements related to the encapsulation technology (see Section 4.1) are those included in lines 4-11. They provide information to select the type of encapsulation. Lines 12-14 are discussed in Section 4.2 and contain aspects of configuration to decide the infrastructure where the service will be deployed. Finally, lines 15-19 are discussed in Section 4.3 and provide information about the requirements that depend on the service's location.

Listing 1.2 presents the output generated by our proposed solution after processing the service requirements. The resulting *deploymentSpecification.yml* file includes the recommended values for the encapsulation type ($V_E$), infrastructure type ($V_I$), and location ($V_L$). In this case, these values indicate a deployment in a lightweight virtualization environment hosted on a VPS server located at the network edge. The options associated with $V_E$, $V_L$, and $V_I$ are fully independent from one another; no constraints exist between them. This independence enables a large number of valid combinations, allowing the generation of highly tailored deployment solutions.

Variable $V_E \in \{NV, HV, LV\}$ is related to the type of encapsulation of the service. When $V_E$ is *NV*, it means that it is not recommended to encapsulate the service. In the case that $V_E$ is *HV* or *LV*, the service should be encapsulated with heavy virtualization (HV)

or light virtualization (LV), respectively. $V_I \in \{VPS, CC\}$ indicates the type of infrastructure in which the service is deployed. If $V_I$ is *VPS* it means that it is recommended to deploy the service in a virtual private service (VPS) infrastructure, while $V_I = CC$ means that it is recommended to deploy the service in a cloud computing (CC) infrastructure. Finally, $V_L \in \{CC, EC\}$ indicates the location most recommended to deploy the service, $CC$ meaning a centralized infrastructure (cloud computing) and $EC$ a non-centralized infrastructure (edge computing).

To verify the results obtained by the recommendation mechanisms, a prototype has been developed and deployed on two different infrastructures to compare the performance. The prototype and its operation are described in Section 5.1.

### 5.1. Target Scenario Emulation

This section presents the test environment employed and the main components used by the developed prototype. Fig. 3 shows a diagram of the location and connection of the components. The test environment attempts to simulate a multimedia content distribution scenario. The scenario models two multimedia content distribution options, the first option (option (a)) simulates a multimedia content distribution system deployed on a cloud computing provider where users connect to this server to consume multimedia content. The second option (option (b)) also emulates a content distribution system, but in this case, the streaming server is hosted in a data center with less computing power, in a network segment close to the users who consume the content. There are obvious differences between the systems in options. The streaming server in option (a) has a higher processing capacity than the server deployed in option (b), so option (a) will be able to process a greater number of requests and could serve more users. However, option (a), located somewhere on the Internet, could present worse network characteristics compared to the connections used in option (b). For example, the latency and packet loss parameters are expected to be worse in option (a) compared to option (b).

### 5.2. Performance Analysis

This section presents a comparison of the operation of a distribution system in different configurations. To simulate the operation as realistically as possible, we have used a video streaming server that provides videos on demand. Several tests have been carried out in which a set of nodes plays content from the server at a given moment. Several sets of nodes have been defined, including 25, 50, 100, 150, and 200 nodes.

The purpose of the tests is to verify how the end-user experience may be affected when trying to play multimedia content on a video distribution system. In theory, the video distribution service can vary the quality of service depending on whether the allocated resources are capable of meeting its needs. The processing capacity of the video streaming server is crucial to ensure its proper functioning. As the number of requests the server receives increases, there is a risk of overload. As a consequence, the time to content delivery is probably going to increase. Another relevant resource is related to the network when the server connects to the clients, since network saturation and bandwidth limitation are key factors for proper operation. The video distribution service can vary the quality of the service depending on whether the allocated resources can meet its needs. Among the

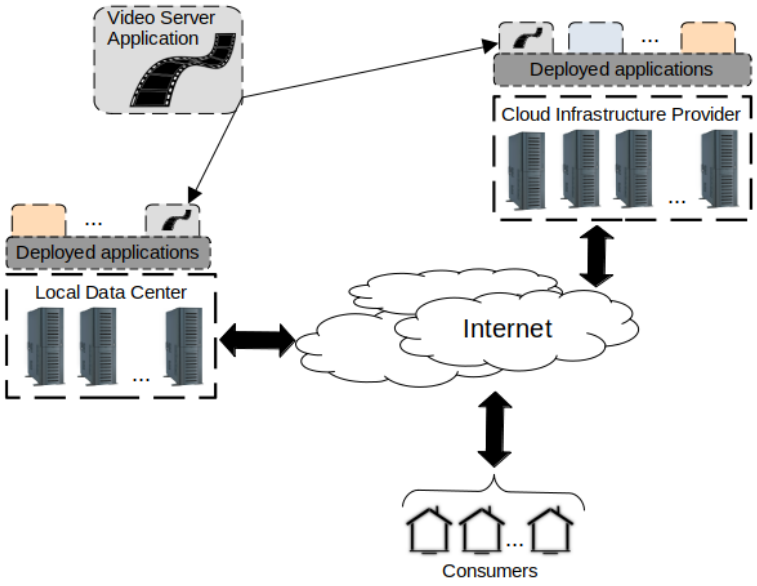



**Fig. 3.** A scenario consisting of a media server deployed in the cloud or on the edge network that delivers videos to consumers

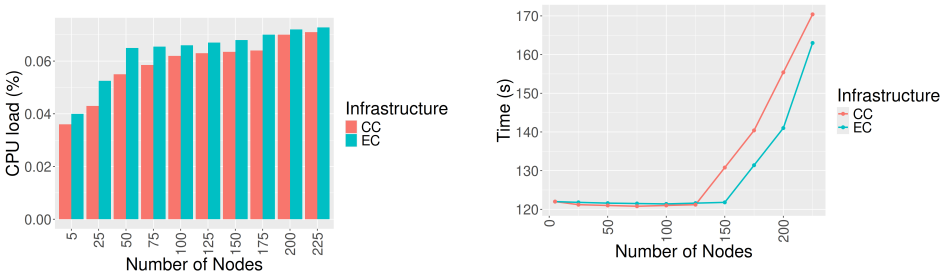
resources affected, one of the most important is the processing capacity available to the video streaming server, if it receives many requests it will need to process them all and will produce delays and latencies in the delivery of content. Another relevant resource is related to the network that connects the server to the clients, since network saturation and bandwidth limitation are key factors for proper operation.

Fig. 3 shows two possible options in which consumers view a video served on a media server. In option (a) the media server is deployed in a Cloud Computing infrastructure (CC). Option (b) presents the same case, but the media server is hosted in a data centre deployed at the edge of the network, in an Edge Computing infrastructure (EC). To model these options, we have developed a simulator based on Docker containers because they allow us to easily deploy the services involved and configure functional parameters that make it easy to simulate both options (a) and (b). The simulator consists of a set of containers, one container acting as a video server and the other containers simulating clients that consume the videos provided by the video server. Two networks, *Network1* and *Network2*, have also been created to simulate the path of the packets until they reach the video server. *Network1* and *Network2* are configured with different parameters that affect the performance of their connections, namely latency, bandwidth, and packet loss rate. In particular, *Network1* configuration is similar to that that connects consumers to a content server deployed in the Cloud Computing Infrastructure (CC), shown in option (a). While *Network2* configuration emulate to a Edge Computing infrastructure (EC), represented in option (b), where the media server is deployed at the edge of the network. The network parameters are set as follows: *Network1*: latency=15ms, bandwidth=50Mbps and packets loss rate=5%; *Network2*: latency=7ms, bandwidth=300Mbps and packets loss rate=2.5%.

To differentiate one option from the other, we have also considered reducing the computational resources available in the containers on which the media server runs. In option (b), the CPU resource of the container in which the media server is deployed has been reduced to 50% with respect to option (a) because we assume that the computational capacity in a datacenter at the edge of the network is more limited than in the cloud.

For a more accurate simulation, it is also important to adjust the computational resources to analyze the behavior of the media server when it is deployed in the cloud and at the edge of the network. Adequately satisfying the computational resources can significantly influence the response time of requests made to the video server by consuming nodes. If the server has abundant computational resources, the server could comfortably meet the demand. Otherwise, the server would have problems meeting requests and could even stop responding.

The results presented in this section are the average of 5 experiment repetitions.



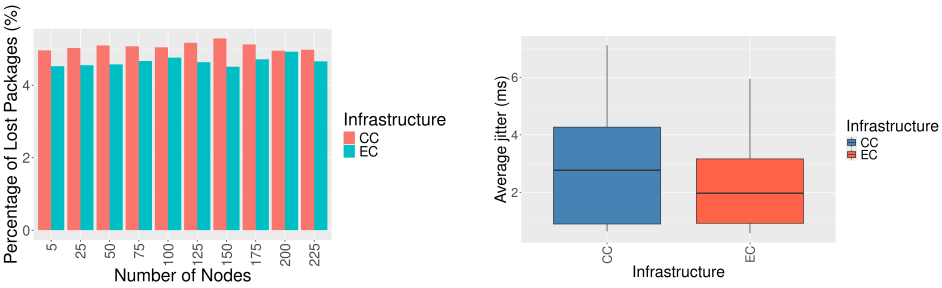

(a) Server CPU load

(b) Average time spent viewing multimedia content

**Fig. 4.** (a) Server CPU load (b) Average time spent viewing multimedia content

Fig. 4 (a) shows the processor load depending on the number of nodes. In scenario (a) (CC, Cloud Computing), the CPU load gradually increases as the number of clients increases. In scenario (b) (EC, Edge Computing), the CPU load is higher and grows faster than in scenario (a). The use of Edge Computing brings the processing closer to the user, however, the Edge Computing infrastructure has fewer resources compared to cloud-based infrastructures. This can lead to higher CPU stress on the Edge Computing infrastructure while the number of connected clients increases. In the cloud infrastructure, this does not occur because of its scalability, which allows it to handle more users with less impact on the CPU.

Fig. 4 (b) shows the average display time of the content as a dependency of the number of connected clients. In both options, the display time is initially stable, adjusting to the duration of the video (120 s), but when the load increases, the system collapses at a critical point, shooting up the display time. The collapse occurs earlier in scenario (a), possibly because the path that the packets have to travel to reach the destinations may be a bottleneck at some point in the network, consequently degrading the quality of service. In the case of the edge computing option, since the server is closer to consumers, network traffic must travel a shorter path, reducing the factors that can affect traffic transit.




(a) Comparison of the percentage of lost packets between cloud computing and edge computing infrastructures

(b) Comparison of average jitter measured between cloud computing and edge computing infrastructures

**Fig. 5.** (a) Comparison of the percentage of lost packets between cloud computing and edge computing infrastructures (b) Comparison of average jitter measured between cloud computing and edge computing infrastructures

In terms of network transmission quality, there are significant differences. The percentage of packet loss (Fig. 5 (a)) is lower in EC than in CC under high concurrency conditions, which directly results in more stable content viewing. Packet loss is a critical metric for measuring service quality and occurs when packets sent by the server do not reach the client, due to network congestion, node saturation, or transmission errors. In multimedia applications, even low packet loss can be perceived as dropouts, image degradation, or interruptions in media content display. A particularly relevant parameter in multimedia transmission applications is jitter, defined as the variation in packet arrival times on the network. Low jitter indicates regular and predictable delivery, which translates into a satisfactory user experience. However, high jitter causes irregularities that degrade the user experience. Fig. 5 (b) shows that EC maintains low jitter even under high load conditions, while CC shows a significant increase as concurrency increases. This confirms that EC provides greater temporal stability, a key factor in services sensitive to variability, such as real-time streaming.

The latency analysis also reinforces this conclusion. Latency is the total time it takes for a packet to travel from the source (server) to the destination (client). This is a fundamental parameter as it directly affects the speed with which users receive information. In multimedia services, low latency values improve user experience, while high values can cause noticeable delays. Fig. 6 shows the minimum latency values, where the EC consistently shows shorter times due to the proximity of the nodes to the end user. In contrast, the maximum latency (also shown in Figure 6) reflects that CC experiences much higher peaks when concurrency increases, while EC, although it also increases its latency in extreme conditions, does so in a more contained manner. This means that the edge infrastructure not only reduces average response times, but also limits extreme peaks that can compromise service stability.

The lower curves in Fig. 6 also show the response times (RTT) standard deviation. The standard deviation acts as a complementary indicator to jitter. Although jitter reflects variations in arrival intervals between consecutive packets, the standard deviation measures the overall dispersion of response times from the average. A low value indicates a more

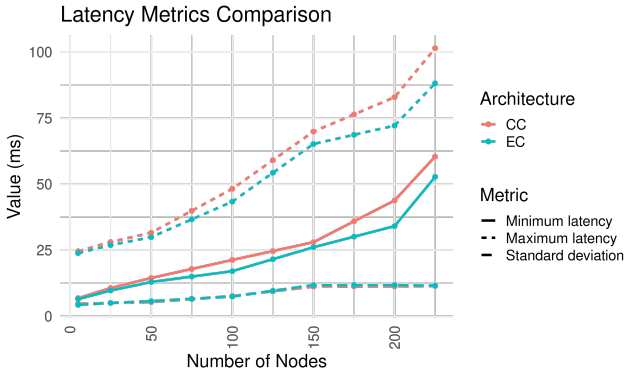



**Fig. 6.** Comparison of maximum and minimum latency measured between cloud computing infrastructures and edge computing versus standard network latency

stable and predictable network, whereas a high value reveals instability, with significant fluctuations affecting service consistency. Values between 1–5 ms are adequate for most interactive and multimedia applications. Values between 5–10 ms may be sufficient for video-on-demand services, but variations are already noticeable in delay-sensitive applications. Values greater than 10 ms present a risk of micro-cuts, buffering, or loss of fluidity in real-time multimedia services. As shown in Fig. 6 EC achieves lower standard deviations compared to CC, resulting in greater consistency in communication and therefore a more satisfactory user experience. In combination with jitter values, this metric confirms that systems deployed on the edge offer more robust performance under intensive loads.

In summary, the experimental comparison shows that Edge Computing outperforms Cloud Computing in terms of scalability, reliability, and network stability. Lower packet loss, lower jitter, reduced latency, and lower standard deviations position EC as an ideal infrastructure for streaming applications and interactive services where quality of experience is a critical factor.

### 5.3. Discussion

The objective of developing the prototype presented in Section 5 was to evaluate the performance of a video service. The features are listed in Listing 1.1 . Listing 1.2 in Section 5 shows the result of the selection process proposed in this work, which provides the recommended values for encapsulation type ($V_E$), infrastructure type ($V_I$), and location ($V_L$).

In the case of the location selection process, the satisfaction of characteristics 15 to 19 coincides with the decision to use EC. As can be seen in the analysis in the previous section, the characteristics that indicate latency (line 15) and jitter (line 16) are critical parameters in multimedia transmission, and the results confirm the advantage of EC. While CC shows high peaks of maximum latency (Fig. 6) and a notable increase in jitter under high concurrency (Fig. 5 (b)), in EC both parameters remain low and stable, ensuring higher quality of experience for the user.

The characteristics *distanceToUser* (line 17), *locationAwareness* (line 18), and *centralizationDegree* (line 19) are factors closely related to location. Centralization in remote data centers increases dependence on the transport network, negatively affecting latency and stability. In contrast, the geographical proximity of the nodes in EC reduces the distance perceived by the user and disperses the load, mitigating centralized saturation.

For the choice of encapsulation technology, the algorithm recommends using lightweight virtualization, for example Docker, which is what has been used in the prototype. Docker meets the requirements of fast deployment and startup (lines 5 and 6) because containers share the host operating system kernel and start up in seconds. However, VMs require a complete guest operating system startup, increasing boot time; while the option no virtualization, deploying the same software contained in the container but installed directly on the host machine, is even slower and more difficult to automate. Another key aspect is portability and compatibility. The definition of the portability attribute (line 8) shows that the service must be relatively easy to transfer. Docker encapsulates dependencies and execution environments, which reduces problems derived from limited compatibility (line 7) and facilitates its execution in heterogeneous infrastructures such as those of Edge Computing.

The high resource consumption attribute (line 10) makes the choice of virtualization particularly relevant. Docker introduces minimal overhead compared to VMs, which require additional resources to maintain complete operating systems. Thus, the density of instances per node is higher in Docker, allowing it to support more concurrent clients without degrading performance, which aligns with the high scalability (line 12) defined for the service. In addition, the ease of management (line 11) is another advantage of Docker because it offers standardized orchestration, monitoring, and update mechanisms, which simplifies the operation of distributed services.

## 6. Conclusion

Typically, automated service deployment mechanisms are modeled using Integer Non-Linear Programming (INLP) problems. However, it is not a viable solution due to the NP-hard nature of INLP problems, particularly when dealing with large-scale systems or those with a multitude of factors to consider. As a result, most of the work in this field relies on heuristic-based algorithms to efficiently solve problems within acceptable time frames. Our proposed solution seeks to address this issue by providing the capability to define features and requirements that help reduce the search space compared to existing approaches. It is important to note that, unlike other approaches, our solution does not focus on optimizing the performance of services or applications. Instead, it focuses on creating optimal configurations for software deployment, thereby enhancing compatibility, management, and the deployment process of applications and services.

This work proposes an architecture to improve the deployment of services on different infrastructures. The solution also proposes three selection mechanisms that help select the most appropriate configuration for the deployment of a service. For this purpose, the following configuration selection processes have been presented: the selection process to decide the most appropriate encapsulation technology with which to deploy a service; the selection process to detect the appropriate infrastructure in which to deploy the service; and the selection process to choose the most appropriate location in which to deploy the

service. To carry out these processes, a set of relevant service requirements have been identified and, in addition, a set of characteristics related to the configuration option on which the services are deployed have also been identified. The work detects which requirements are satisfied by the characteristics associated with the different configuration options available and proposes a deployment configuration.

In summary, our proposal presents a valuable and efficient alternative to INLP-based approaches in service deployment, effectively tackling the complexities of deploying applications and services in large-scale systems by focusing on optimal software configuration for enhanced compatibility, management, and deployment. For future research, we plan to analyze additional service deployment scenarios to identify a wider range of relevant requirements. Furthermore, we intend to explore the incorporation of more features to enable a more granular definition of infrastructures and address other complex configuration aspects.

**Acknowledgments.** Iván Bernabé Sánchez has been funded by the Spanish Ministry of Universities through a grant related to the Requalification of the Spanish University System 2021–23 Margarita Salas by the Carlos III University of Madrid. This work was supported by the Spanish Government under the research projects "Enhancing Communication Protocols with Machine Learning while Protecting Sensitive Data (COMPROMISE)" PID2020-113795RBC32, COSASS PID2021-123673OB-C32, funded by MCIN/AEI/10.13039/501100011033 and by "ERDF A way of making Europe", and VAE: TED2021-131295B-C33 funded by MCIN/AEI/ 10.13039/501100011033 and by the "European Union NextGenerationEU/PRTR". This research also results from the I-Shaper Strategic Project (C114/23), due to the collaboration agreement signed between the Instituto Nacional de Ciberseguridad (INCIBE) and the Universidad Carlos III de Madrid. It is worth highlighting that this initiative is being carried out within the framework of the Recovery, Transformation and Resilience Plan funds, funded by the European Union (Next Generation).

## References

1. Agarwal, S., Malandrino, F., Chiasserini, C.F., De, S.: Vnf placement and resource allocation for the support of vertical services in 5g networks. IEEE/ACM Transactions on Networking 27(1), 433–446 (2019)
2. Alnaim, A.K., Alwakeel, A.M., Fernandez, E.B.: A pattern for an nfv virtual machine environment. In: 2019 IEEE International Systems Conference (SysCon). pp. 1–6 (2019)
3. Bari, F., Chowdhury, S.R., Ahmed, R., Boutaba, R., Duarte, O.C.M.B.: Orchestrating virtualized network functions. IEEE Transactions on Network and Service Management 13(4), 725–739 (2016)
4. Battisti, A.L.É., Macedo, E.L.C., Josué, M.I.P., Barbalho, H., Delicato, F.C., Muchaluat-Saade, D.C., Pires, P.F., Mattos, D.P.d., Oliveira, A.C.B.d.: A novel strategy for vnf placement in edge computing environments. Future Internet 14(12), 361 (2022)
5. Cziva, R., Pezaros, D.P.: Container network functions: Bringing nfv to the network edge. IEEE Communications Magazine 55(6), 24–31 (2017)
6. Debeau, E., Quintuna-Rodriguez, V.: Onap: An open source toolkit for zero touch automation. In: Design Innovation and Network Architecture for the Future Internet, pp. 212–249. IGI Global (2021)
7. Dräxler, S., Karl, H., Peuster, M., Kouchaksaraei, H.R., Bredel, M., Lessmann, J., Soenen, T., Tavernier, W., Mendel-Brin, S., Xilouris, G.: Sonata: Service programming and orchestration for virtualized software networks. In: 2017 IEEE international conference on communications workshops (ICC workshops). pp. 973–978. IEEE (2017)

8. Ericsson: Ericsson mobility report. Tech. rep., Ericsson (Nov 2024), https://www.ericsson.com/4adb7e/assets/local/reports-papers/mobility-report/documents/2024/ericsson-mobility-report-november-2024.pdf, accedido: 15 de diciembre de 2024
9. Fei, X., Liu, F., Xu, H., Jin, H.: Towards load-balanced vnf assignment in geo-distributed nfv infrastructure. In: 2017 IEEE/ACM 25th International Symposium on Quality of Service (IWQoS). pp. 1–10 (2017)
10. Gangadharan, G.: Open source solutions for cloud computing. Computer 50(01), 66–70 (2017)
11. Gao, X., Liu, R., Kaushik, A., Zhang, H.: Dynamic resource allocation for virtual network function placement in satellite edge clouds. IEEE Transactions on Network Science and Engineering 9(4), 2252–2265 (2022)
12. Giust, F., Costa-Perez, X., Reznik, A.: Multi-access edge computing: An overview of etsi mec isg. IEEE 5G Tech Focus 1(4), 4 (2017)
13. Goudarzi, S., Soleymani, S.A., Anisi, M.H., Jindal, A., Dinmohammadi, F., Xiao, P.: Sustainable edge node computing deployments in distributed manufacturing systems. IEEE Transactions on Consumer Electronics pp. 1–1 (2023)
14. Grinter, R.E.: Using a configuration management tool to coordinate software development. In: Proceedings of conference on Organizational computing systems. pp. 168–177 (1995)
15. Gu, L., Tao, S., Zeng, D., Jin, H.: Communication cost efficient virtualized network function placement for big data processing. In: 2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). pp. 604–609. IEEE (2016)
16. Harris, D., Raz, D.: Dynamic vnf placement in 5g edge nodes. In: 2022 IEEE 8th International Conference on Network Softwarization (NetSoft). pp. 216–224. IEEE (2022)
17. Hintsch, J., Görling, C., Turowski, K.: A review of the literature on configuration management tools (2016)
18. Jha, D.N., Li, Y., Jayaraman, P.P., Garg, S., Ushaw, G., Morgan, G., Ranjan, R.: Challenges in deployment and configuration management in cyber physical system. Handbook of Integration of Cloud Computing, Cyber Physical Systems and Internet of Things pp. 215–235 (2020)
19. Jin, P., Fei, X., Zhang, Q., Liu, F., Li, B.: Latency-aware vnf chain deployment with efficient resource reuse at network edge. In: IEEE INFOCOM 2020 - IEEE Conference on Computer Communications. pp. 267–276 (2020)
20. Kim, S.I., Kim, H.S.: A vnf placement method based on vnf characteristics. In: 2021 International Conference on Information Networking (ICOIN). pp. 864–869. IEEE (2021)
21. Kourtis, M.A., McGrath, M.J., Gardikis, G., Xilouris, G., Riccobene, V., Papadimitriou, P., Trouva, E., Liberati, F., Trubian, M., Batallé, J., et al.: T-nova: An open-source mano stack for nfv infrastructures. IEEE Transactions on Network and Service Management 14(3), 586–602 (2017)
22. Koziolek, H., Hark, R., Eskandani, N., Nguyen, P.S., Rodriguez, P.: Tosca for microservice deployment in distributed control systems: Experiences and lessons learned. In: 2023 IEEE 20th International Conference on Software Architecture Companion (ICSA-C). pp. 11–21. IEEE (2023)
23. Kumar, M., Walia, G.K., Shingare, H., Singh, S., Gill, S.S.: Ai-based sustainable and intelligent offloading framework for iiot in collaborative cloud-fog environments. IEEE Transactions on Consumer Electronics pp. 1–1 (2023)
24. Leivadeas, A., Kesidis, G., Ibnkahla, M., Lambadaris, I.: Vnf placement optimization at the edge and cloud. Future Internet 11(3), 69 (2019)
25. Li, L., Qiu, Q., Xiao, Z., Lin, Q., Gu, J., Ming, Z.: A two-stage hybrid multi-objective optimization evolutionary algorithm for computing offloading in sustainable edge computing. IEEE Transactions on Consumer Electronics pp. 1–1 (2024)
26. Mandal, P.: Comparison of placement variants of virtual network functions from availability and reliability perspective. IEEE Transactions on Network and Service Management 19(2), 860–874 (2022)

27. Pham, C., Tran, N.H., Ren, S., Saad, W., Hong, C.S.: Traffic-aware and energy-efficient vnf placement for service chaining: Joint sampling and matching approach. IEEE Transactions on Services Computing 13(1), 172–185 (2017)
28. Promwongsa, N., Ebrahimzadeh, A., Glitho, R.H., Crespi, N.: Joint vnf placement and scheduling for latency-sensitive services. IEEE Transactions on Network Science and Engineering 9(4), 2432–2449 (2022)
29. Reddy, K.H.K., Goswami, R.S., Luhach, A.K., Chatterjee, P., AlNumay, M., Roy, D.S.: Eflsm:-an intelligent resource manager for fog layer service management in smart cities. IEEE Transactions on Consumer Electronics pp. 1–1 (2024)
30. Reid, A., González, A., Armengol, A.E., de Blas, G.G., Xie, M., Grønsund, P., Willis, P., Eardley, P., Salguero, F.J.R.: Osm scope, functionality, operation and integration guidelines. ETSI, White Paper (2019)
31. Sallam, G., Ji, B.: Joint placement and allocation of vnf nodes with budget and capacity constraints. IEEE/ACM Transactions on Networking 29(3), 1238–1251 (2021)
32. Soltesz, S., Pötzl, H., Fiuczynski, M.E., Bavier, A., Peterson, L.: Container-based operating system virtualization: a scalable, high-performance alternative to hypervisors. In: Proceedings of the 2Nd ACM SIGOPS/EuroSys european conference on computer systems 2007. pp. 275–287 (2007)
33. Sriram, G.: Edge computing vs. cloud computing: an overview of big data challenges and opportunities for large enterprises. International Research Journal of Modernization in Engineering Technology and Science 4(1), 1331–1337 (2022)
34. Sun, J., Liu, F., Wang, H., Wu, D.O.: Joint vnf placement, cpu allocation, and flow routing for traffic changes. IEEE Internet of Things Journal 10(2), 1208–1222 (2022)
35. Virtualisation, N.F.: An introduction, benefits, enablers, challenges & call for action. In: White Paper, SDN and OpenFlow World Congress. p. 73 (2012)
36. Wu, Y., Tang, S., Zhang, L., Fan, L., Lei, X., Chen, X.: Resilient machine learning based semantic-aware mec networks for sustainable next-g consumer electronics. IEEE Transactions on Consumer Electronics pp. 1–1 (2023)
37. Zeng, Z., Xia, Z., Zhang, X., He, Y.: Sfc design and vnf placement based on traffic volume scaling and vnf dependency in 5g networks. CMES-Computer Modeling in Engineering & Sciences 134(3) (2023)

**Iván Bernabé-Sánchez** received a degree in information technology engineering from the Carlos III University of Madrid, Leganés, Spain, in 2007 and a PhD degree in 2021. His research interests include the virtualization of devices and infrastructures defined through code, cloud and edge computing architectures, and self-configuration systems mechanisms based on knowledge representation and semantic technologies. He has participated in several nationally or internationally funded research projects.

**Daniel Diaz-Sánchez** is a full professor at University Carlos III of Madrid. His research interests include distributed authentication/authorization, content protection, distributed computing, fog computing, IoT and Smart Cities. Díaz-Sánchez received a PhD in telematics engineering from UC3M.

**Alberto Fernández** is a full professor at the University Rey Juan Carlos (URJC) in Madrid, where he is a member of the Artificial Intelligence Group of the CETINIA research centre. He obtained a PhD in Computer Science from the URJC. His main research lines are multi-agent systems, knowledge representation, semantic technologies,

open systems, etc. He is especially interested in the application of previous technologies in domains such as intelligent transportation systems, fleet management, etc. He has participated in many national and international projects on the above topics and has published more than 80 articles in international journals, books and conferences.

**Holger Billhardt** received his M.Sc. in computer science from the TH Leipzig, Germany, and his PhD in computer science at the Universidad Politécnica in Madrid. He is currently a full professor of computer science at Universidad Rey Juan Carlos in Madrid, where he is a member of the Artificial Intelligence Group at the Centre for Intelligent Information Technologies (CETINIA). His research is concerned with multi-agent systems, especially with the coordination of agents in distributed, open and dynamic environments. He is the author or co-author of more than 100 research papers and has participated in several nationally or internationally funded research projects.

**Sascha Ossowski** is a full professor of computer science and director of the CETINIA research centre at the University Rey Juan Carlos in Madrid. He received a MSc degree in informatics from U Oldenburg (Germany) and a PhD in artificial intelligence from TU Madrid (Spain). The main themes of his research refer to models and mechanisms for coordination in all sorts of agent systems and environments. He was co-founder of the European Association for Multiagent Systems (EURAMAS), chaired the European COST Action on Agreement Technologies, and is an emeritus board member of the International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS).

*Received: September 15, 2025; Accepted: November 30, 2025.*