Specification and Unattended Deployment of Home Networks at the Edge of the Network

Iván Bernabé-Sánchez, Daniel Díaz-Sánchez, *Senior Member, IEEE*, and Mario Muñoz-Organero, *Member, IEEE*

***Abstract*—Consumer devices continue to expand their capabilities by connecting to digital services and other devices to form information-sharing ecosystems. This is complex and requires meeting connection requirements and minimal processing capabilities to ensure communication. The emergence of new services, and the evolution of current technologies, constantly redefine the rules of the game by opening up new possibilities and increasing competition among service providers. Paradigms such as edge computing, softwarization of physical devices, self-configuration mechanisms, definition of software as a code and interoperability between devices, define design principles to be taken into account in future service infrastructures. This work analyzes these principles and presents a programmable architecture in which services and virtual devices are instantiated in any computing infrastructure, as cloud or edge computing, upon request according to the needs specified by service providers or users. Considering that the target computing infrastructures are heterogeneous, the solution defines network elements and provides network templates to ensure it can be deployed on different infrastructures irrespectively of the vendor. A prototype has been developed and tested on a virtualized cloud-based home network relying on open source solutions.**

***Index Terms*—Connected consumer devices, fog computing, orchestrator.**

I. Introduction

The ECOSYSTEM of digital services available for consumer devices has been growing for years. For instance, video streaming has continued to grow and it is expected its traffic will reach 78% of the total traffic consumed by Internet consumers in 2021 [1]. That growth is not only caused by the significant increase of the available digital content, but also by the increase of the devices connected to the Internet that are able to consume these contents, considering also mobile devices and other personal consumer devices are becoming the preferred clients [1]. Currently, most services targeting heterogeneous devices are Over-the-top (OTT) services so data is sent to the user bypassing traditional mediums. OTT services allow multiple devices to consume the same contents since there is no need for an underlying service infrastructure. However, these services would perform better using dedicated infrastructure providing an increased user experience whereas requiring less resources. Continuing with the video streaming example, multicast and distributed caches can considerably reduce network bandwidth at the backbone and improve video quality. In a future scenario, characterized by a great diversity of devices, many potential users and available services, it would be convenient to rely on infrastructure and to combine different transmission mechanisms in a per-service basis. Since physical infrastructure can be expensive to deploy, it is sensible to consider this digital service ecosystem would require using dynamic and scalable infrastructures. Fortunately, Cloud Computing paradigm, and its different embodiments as Edge Computing and Fog Computing, provide such an infrastructure: capable of dynamically instantiate services and network elements thus, coping with the aforementioned requirements whereas reducing investment and operation costs. Nevertheless, the Cloud Computing paradigm has several problems that require special attention as privacy and data Lock-In (or Vendor Lock-In) [2]. Concerning privacy, new mechanisms should be developed that ensure user data is not misused once uploaded to the cloud. The problem of Data Lock-In is specially worrying with regard to the dynamic infrastructure deployment. Nowadays, cloud computing orchestrators are quite heterogeneous and lack of reliable standard interfaces for automated deployment. Thus, an infrastructure definition designed for a given cloud framework is more than likely non instantiable in others.

In spite of the advantages of cloud computing, the increasing number of virtualized devices in the cloud complicates their management. Thus flexible configuration mechanisms, advanced management, decentralized processing, and self-organization are required. The distribution of computing between the cloud and the devices allows moving processing tasks to more appropriate processing infrastructures. In such a way, tasks demanding high computing resources are offloaded to the cloud, keeping basic services, such as graphical user interface (GUI), on the devices that connect to those services. This model requires consumer devices with less computational resources and therefore devices can be cheaper and smaller, reducing transport costs and energy consumption (since complex tasks are offloaded to the cloud). Also, turning devices into more simple versions, will not require complex software to control them and therefore will be easier to perform common tasks as updating or maintaining them; they will be more

Manuscript received November 9, 2019; revised April 8, 2020, June 15, 2020, and August 12, 2020; accepted August 13, 2020. Date of publication August 21, 2020; date of current version November 25, 2020. This work was supported in part by the Project MAGOS under Grant TEC2017-84197-C4-1-R; in part by the Comunidad de Madrid (Spain) through the Project CYNAMON under Grant P2018/TCS-4566; and in part by the European Structural Funds (ESF and FEDER). *(Corresponding author: Iván Bernabé-Sánchez.)*

The authors are with the Telematic Engineering Department, Carlos III University of Madrid, 28911 Leganés, Spain (e-mail: ibernabe@gmail.com; dds@it.uc3m.es; munozm@it.uc3m.es).

Digital Object Identifier 10.1109/TCE.2020.3018543

secure; and development times will reduced. In this article, the authors propose a framework that extends the network functions defined in the specifications and standards of NFV [3] for incorporating higher-level features. The goal is to extend the virtualization of network functions with services instantiation in consumer devices. Authors propose to extend the idea of NFV instead of proposing a new design because NFV has a great acceptance in the industry and so we will contribute by expanding the NFV architecture. In this way, service providers will have additional control of the system software by moving some services, traditionally located on the devices, to the cloud. However, they have to face the problem of deploying services over multiple platforms. To overcome this, we propose a configuration structure to declare components, connections and configurations, in a way it provides the ability to instantiate a given set of services in many different infrastructures. So that this configuration can be applied at various levels in the infrastructures, this work proposes to use a model segmented by software layers that can be managed through templates, configurations and interfaces blocks. Thus, orchestrators can deploy and interconnect software parts in a simple way regardless the environment vendor. The software layer model also includes the use of virtualization mechanisms to ensure software deployments over multiple infrastructures. Self-configuration is also interesting for this work since we propose introducing the concept of virtual ecosystem into domestic environments and these environments require many and complex configuration tasks. Using automatic software management mechanisms guarantees the correct configuration and duplication of services on different platforms. It also speeds up the process. Not using these mechanisms could take more time to complete each process and make configuration mistakes.

The rest of the article is divided into the following sections: Section II analyzes the relevant conceptual changes in service architectures, Section III contains the state of the art, Section IV shows the proposed architecture, Section V presents results, and Section VI comments on the conclusions of the article.

II. Problem Domain

This section describes the problem of orchestrating virtual infrastructure in the aforementioned scenarios; which are the limits of current solutions; what changes may alleviate the problem in future service architectures. To better understand the limitations and proposed solutions, let define the user roles involved in such scenarios. Usually in this context, exist the figure of the cloud provider (CP) which provides infrastructure to service providers (SPs). SPs host their services on CP infrastructure. SPs use software components developed by software manufacturers (SM) to compose services and then the SPs sell those services to service customers (SC) which are consumers of services offered by SPs.

A. Infrastructure Definition

Device softwarization is becoming more important with time. SPs have not only migrated a large part of their services to the cloud, but also have begun to migrate applications to the cloud which were designed to work on smart physical devices. Consumer devices, such as smartphones, tablets, smart TVs, etc., currently contains a large amount of software that can easily be moved to the cloud to improve service quality. In addition, an emerging trend is to move the functionality of physical devices to the cloud to take advantage of the benefits of the cloud as well as extend the capabilities of devices by reducing maintenance and energy costs [4]. Besides the virtualization of devices, it is also possible to replicate home networks in a transparent way. So, virtual applications previously instantiated on user premises will behave in the same way once virtualized. Thus, the broadcast domain and latency of the virtual network segments mimic the behavior of a traditional wired/wireless consumer network. We call it Virtual Home Personal Network (VHPN).

VHPNs provide mechanisms for exchanging information relevant using standard layer-2 protocols. VHPNs are composed of a virtual ecosystem similar the physical world where users interact with software applications in a similar way as they do with traditionally home networks, whereas SPs are able to reduce CAPEX and OPEX.

Virtual consumer ecosystems of networks and services are of paramount importance to define a new generation of value-added services. However, they are complex to configure, deploy, test and operate. Due to that, it is required to define a mechanism to simplify these tasks as much as possible, and the concept of Infrastructure as Code (IaC) can be one of these mechanisms. IaC is intended to specify a given infrastructure and automate its deployment [5]. Several embodiments of IaC rely on domain-specific languages (DSL) to define infrastructure thus, a given ecosystem, containing networks, devices and services, could be recreated on several cloud environments.

However, although there are multiple alternatives proving such domain-specific languages that define infrastructure compatible with several cloud platforms, the variety of IaC initiatives require developers to have a good knowledge of the target cloud infrastructures. The work presented in this article, proposes a generic framework compatible with many cloud platforms to alleviate this problem.

B. Interface Standarization

SPs rely on a set of basic services running in the backend that are needed for the operation of contracted services (back office). Currently, these basic services can be composed of a set (of tens or hundreds) of independent microservices responsible for the operation of a part of the business service. Usually, these services are encapsulated in a service layer controlled from an interface that abstracts the complexity of the internal operation and allows interaction with external entities. This set of services is called Operation Support System (OSS) [6] and allows to monitor, control, analyze, and manage of services. In addition, OSS interacts with services responsible for commercial activities, customer relations, billing, etc. Those services make up the Business Support Systems (BSS).

Traditionally, SPs have designed the OSSs in a closed manner to meet only a particular purpose and without taking into account standard specifications. So, there are proprietary interfaces that limit scalability and interaction with third-party systems and are often monolithic systems that hinder migration to cloud environments. In addition, there are no guidelines on how to integrate new technologies such as SDN and NFV into OSS. It is important since NFV and SDN infrastructures need to dynamically move resources between different points in the network to satisfy variations in traffic. In addition, traditional OSS systems are not dynamic and may not be able to respond to changes in real-time service conditions. To address these issues, the virtualization of OSS and the use of the cloud are the keys, because they increase the flexibility and scalability of the systems while reducing operating costs and also facilitate the mobility of services among cloud vendors.

The industry has worked to create a reference model to interconnect different types of systems. Organizations as the International Telecommunication Union (ITU-T) part of Telecommunications Management Network (TMN) [7] and the ETSI [8] published guidelines to implement OSS/BSS systems. We have analyzed these guidelines and they are only high-level specifications that contain few details for implementation, so its adoption was poor. In addition, we have not found another reference implementation, therefore, the use of these guidelines can lead to integration problems between the parties.

As a solution, two initiatives emerged to solve this problem: Next Generation OSS (NGOSS) by Telemanagement Forum (TMF) and OSS through Java OSS/J. NGOSS defines a framework that can be used for designing of OSS systems. OSS/J offers standards-based interface design Guidelines (APIs OSS/J) for OSS system development.

However these solutions have unresolved issues such as: i) respond to changes/events of the service in real-time; ii) OSS should automatically support both device, static services and network adapter modelling; iii) OSS must be able to work in conjunction with NFV orchestrator and both must interpret the operating policies in the same way because both can modify resources through the NFV infrastructure (NFVI) interfaces. The solution showed in this article proposes an architecture that approaches those needs.

C. System Autoconfiguration

The ability to autoconfigure systems is a very attractive issue for the industry for the benefits it brings. In this work, self-configuration is a concept of special interest because it facilitates the management and maintenance of a multitude of services and systems while drastically reducing the workload of the operators. For us, a virtual ecosystem in the home network is made up of user devices that could interact with each other and with remote services hosted in the cloud forming a large digital ecosystem. This ecosystem requires a large number of components, so it is necessary to have mechanisms with ability of self-configuration and constant adaptation. For this reason, it is necessary to use: flexible architectures, re-usable software blocks instead of specific software, use standards definitions rather than creating new ones, have capabilities for service lifecycle management, and work with open components from multiple vendors instead of vendor-specific solutions.

The industry tries to adapt to these needs and proposes solutions through its standardization bodies (SDO) such as ETSI. ETSI defined the MANO architecture [9] to allow the management and automatic control of services which belong to multiple providers located at different domains. An orchestrator or controller in charge of managing a large system has to deal with a large number of applications and domains which is really complicated. To address the problem of autoconfiguration ONAP uses a divide and conquer strategy [10]. It proposes to decompose large systems into smaller and manageable subsystems distributed over different providers which collaborate for a greater purpose. In other words, it prevents a single orchestrator to manage the resources of all the elements involved in the system by proposing that each subsystem controls the resources of its domain through a local orchestrator. This model requires communication between the different local orchestrators so that they are able to work together. Therefore, it is necessary to have APIs well specified and common information models. ETSI has provided interfaces to MANO but MANO is focused to work with network devices such as switches, firewalls (FWs), Set-Top Box (STB), residential gateway (RGW), etc. but they do not directly contemplate the inclusion of smart devices. NFV has been thinking to work mainly with network devices such as FWs, routers, RGWs, etc. Due to the trend of softwarization of devices and the need to connect services of the Internet with smart devices, the inclusion of new general-purpose devices within the NFV domain is emerging. This article considers these needs and proposes mechanisms to be able to deploy virtualized smart devices in an NFV infrastructure.

D. From Cloud to Edge Computing

The cloud computing paradigm has become the first choice to deploy services and infrastructure. Resource elasticity, low cost and pay-per-use makes cloud computing an ideal ecosystem to ensure cost effectiveness.

In spite of these advantages, the emergence of new digital services, the increase of the necessary computational resources for service and the importance of the user context for the functioning of the services have generated the need to design digital services more personalized. In other words, services require optimizing responsiveness and having information about the user's context. A solution to meet these needs is to move digital services to a physical location close to the customer to improve: i) response times through the reduction of latency and increased throughput; ii) take and provide with local information to applications to adapt themselves to the local context; iii) refine security mechanisms according to the local scope (security policies adapted to the local scope or subject to legal laws of the region) and the consumption of digital services in which information does not flow through the Internet. These requirements fit perfectly with edge computing and SPs aware of those requirements try to bring their services to consumers. In order to maximize the approximation of services to users, SPs may contract third-party data centers on which to deploy services. But that can be a problem because, in an ideal scenario, local data centers will be composed by heterogeneous well-defined interfaces and standard-based management modules that allow the control of services. However, data center providers have their infrastructure, optimized and adapted to their needs, and SPs have to take into account the characteristics of each virtualization platform. For this reason, when they design their services, they have to ensure the operation of the environment of execution on which their services are deployed. This article presents in the following sections an approach to solve the problem of interoperability.

III. Related Work

A. Interoperability and Interfaces

To address the interoperability problem, systems offer interfaces to connect with them from external entities. Section II-B exposes the efforts made by organizations to design a layer of basic OSS/BSS services with standardized interfaces to the communication between the backend of the internal services of a system with external entities and expand or reconfigure them as discussed in Section II-C.

Interoperability requires the management and control of the operation of complex systems through well-defined interfaces. In this direction, ETSI was the first to define a framework for the management and orchestration of NFV services through the MANO-NFV specification [9]. From this specification have emerged implementation [11] efforts as Tacker [12], OPENFV, Open Baton, OPEN-O, OpenMANO and T-NOVA [13].

All these implementations act as a black box encapsulating all the operations necessary for the proper provisioning of the services that they offer. From the point of view of the SPs, these solutions allow consumers to choose those that best suit their needs. However, this situation has a dark side since the migration of a service to another platform can be a problem due to interoperability problems, which implies the execution of manual integration tasks, migration and others. The industry has tried to solve these problems with high-level interfaces based on the OSS/BSS initiative. For years, OSS/BSS systems were developed privately and some standards developing organization (SDO) have tried to create specifications for these interfaces, like the Organizations as the International Telecommunication Union (ITU-T) under the banner of Telecommunication Management Network (TMN) [7] and the ETSI [8]. However, these works present high-level guidelines without defining an API in detail and since the lack of definition of a reference data model makes it impossible to interoperate services between different operators [14]. The MEF Forum has noticed the new needs of the OSS/BSS layer of the current emerging systems (mainly SDN and NFV), and has defined an intermediate layer called Lifecycle Service Orchestration (LSO) [10]. LSO provides a model of abstraction for services, resources and products which hides the complexity of the technologies and network layers associated with applications and services. LSO is currently in the process of certification and inclusion of new products in its catalog of compatible systems and has a long way to go to consolidate its acceptance in the world of the industry.

In conclusion, for communication via interfaces is necessary to use a common data model in addition to have well-defined interfaces. We think that this requirement complicates communication between systems. So, this work proposes to use a model segmented by software layers and also the management of each layer through templates, configurations and interfaces blocks.

B. From Physical Devices to Virtual Devices

Section II-A has commented on the importance of advances in device virtualization. It also exposes how the variety of infrastructure providers poses a compatibility issue because it requires a device definition for each infrastructure and how virtualization of devices tries to alleviate this problem. This approach is not new, the benefits of convergence between communication networks and computer resources are clear [15]. In fact, some works propose approximations to carry functions of CPE devices to the core of the network [16], [17] and [18]. This is motivated by a tendency to move device functions to the cloud [16] and [17]. In [16] and [17] explain the concept of virtualizing devices in the cloud. They propose to move part of the customer premises equipment (CPE) functions to the data center located in the core of the network as a virtual CPE (vCPE). It is based on a minimal hardware device located at the edge of the network while most of the intelligence stays in the cloud. Nevertheless, as time goes by, this concept has gone further and has tried to integrate into the NFV ecosystem [18] where the inclusion of the Home Gateway in a VNF is proposed. So, it may be connected to other VNFs forming service function chain (SFC) [19]. This complies with the guidelines generated by the ETSI. The ETSI has published application scenarios for its NFV model integrated by CE devices [20]. This has been a starting point for other researchers who are investigating classifying services according to their complexity. For example, by moving the complex software pieces to the cloud while the most basic services are executed on the local device [21] and [22]. Other works analyze the most appropriate functions to virtualize [21] and [22]. And other works define operating policies focused on reducing energy expenditure and depending on these policies and operating conditions at a given time, the solution will decide where to deploy the virtualized node [23]. All these contributions indicate a clear interest in the virtualization of devices and the NFV infrastructure. However, we have not found works that propose the inclusion of smart devices on NFV or any kind of mechanism to be able to manage them. Our solution proposes a mechanism with the capacity to work with this type of device without problems.

IV. Solution Architecture

The design principles of the architecture were defined considering the requirements presented in the previous section. This section describes: design principles, architecture proposed and potential limitations of this work.

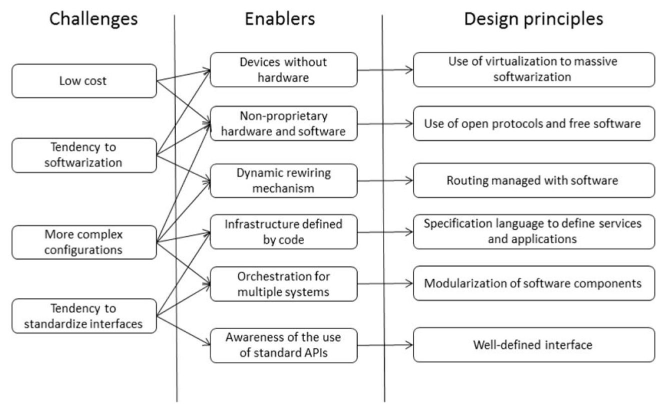
Challenges
Enablers
Design principles
Low cost
Tendency to softwarization
More complex configurations
Tendency to standardize interfaces
Devices without hardware
Non-proprietary hardware and software
Dynamic rewiring mechanism
Infrastructure defined by code
Orchestration for multiple systems
Awareness of the use of standard APIs
Use of virtualization to massive softwarization
Use of open protocols and free software
Routing managed with software
Specification language to define services and applications
Modularization of software components
Well-defined interface

Fig. 1. Challenges, enablers and design principles. Adapted from [24].

A. Principles of Design

In Fig. 1, the enablers are presented to face the challenges presented in the previous section. Based on these enablers the proposed solution has used them as design principles.

1) Devices without hardware. The device virtualization allows adding additional capacities to simple devices reducing capital expenditures (CAPEX).
2) Non-proprietary hardware and software. The proposed solution can be deployed over any virtualized environment avoiding: i) the vendor lock-in problems; ii) use of proprietary firmware installed on devices; iii) limited updates, etc.
3) Dynamic rewiring mechanisms. It uses software-defined network mechanisms allowing dynamically deploying, configuring and controlling networks. This allows to adapt the network according to the traffic and the load of the nodes.
4) Infrastructure defined by code. The specification of network elements through an infrastructure language allows: to independently deploy software pieces on the platform and guarantee the operation regardless of the environment where is executed.
5) Orchestration for multiple systems. Application developed through microservices, modules and software components allows to compose a multitude of services through the orchestration of them.
6) Awareness for the use of standard APIs. Using well-defined interfaces facilitates interconnection with third-party systems. So, it also reduces the time to market, improving interoperability and facilitating the understanding of systems.

B. Architecture Design

Fig. 2 shows the architecture proposed in this article. Our approach allows working with the NFV architecture [25] and with NFV Management and Orchestration (NFV MANO) [9] components. The solution is divided into two main blocks: i) Infrastructure Monitoring and Management system (IMOM); ii) INFrastructure Orchestration and Deployment system (INFORD).

1) Infrastructure Monitoring and Management System: IMOM is a framework which offers to SPs the necessary tools to define, build and deploy service infrastructures. IMOM can be used by SPs to configure their own infrastructure and then deploy services. In addition to service providers, SMs and admin users can also access IMOM to deploy and configure software through the BSS layer. Each of the user profiles has limited access to the platform depending on the role assigned to the user. BSS layer has a user management service located in the Customer Management block which manages user access to IMOM services. CM provides a Web access panel common to all users. When a user is authenticated by the system, a dashboard is displayed with the actions available to the user depending on their role. IMOM provides a catalog of services in the layer called Business Support System (BSS) and configures and deploys necessary components (by means of the INFORD systems) according to the options selected by each SP.

Fig. 2 shows the layers that compose the IMOM architecture. These layers allow SPs to materialize their business models in the form of interconnected services which are deployed using INFORD.

The Business Support Services layer pursues the monetization of the platform via SC information stored, the catalog of available services in the platform and information related to transactions made by SC. SP creates requests of services and software stored in the repository and those requests accepted by Business Support Services layer are then handled by the Services Layer (SL). The SL is in charge to deploy services required, it is composed of the following: Provision Service (PS), Configuration Service (CS) and Monitoring Service (MS). PS and CS together form the orchestrator and are responsible to provide orchestration mechanisms to configure INFORD. The work of the orchestrator is critical because it is responsible for the proper functioning of the services. It takes into account factors such as: the life cycle of each service, the characteristics of each software component and the configuration of the multiple platforms on which the service can be deployed. From this information, the orchestrator generates deployment configuration and execution scripts to facilitate orchestration tasks. In this work, we propose a model segmented by software layers to facilitate the management of services where each layer has a specific mission and works independently. We have identified four layers (see Fig. 3): Domain Network layer, Cluster layer, Node layer and Host layer. The Domain Network Layer (DNL) contains the highest level elements called deployment units (DU). DUs are software containers where virtual smart devices (vSDs), applications and services have been deployed. Each DU is composed of a set of microservices configured to work together to provide the functionality to external elements via its interface. An example of microservices orchestration can be found in the popular microservices container systems (MCS), which have tools capable of managing multiple containers in order to form an application. DNL provides a framework where containers run. DNL contains the VUN described in Section IV-B2. The Cluster layer (CL) provides to DNLs a homogenized execution framework independent of the lower layers. CL also facilitates scalability and provides methods to manage the life cycle of the DUs. The Node layer (NL) has

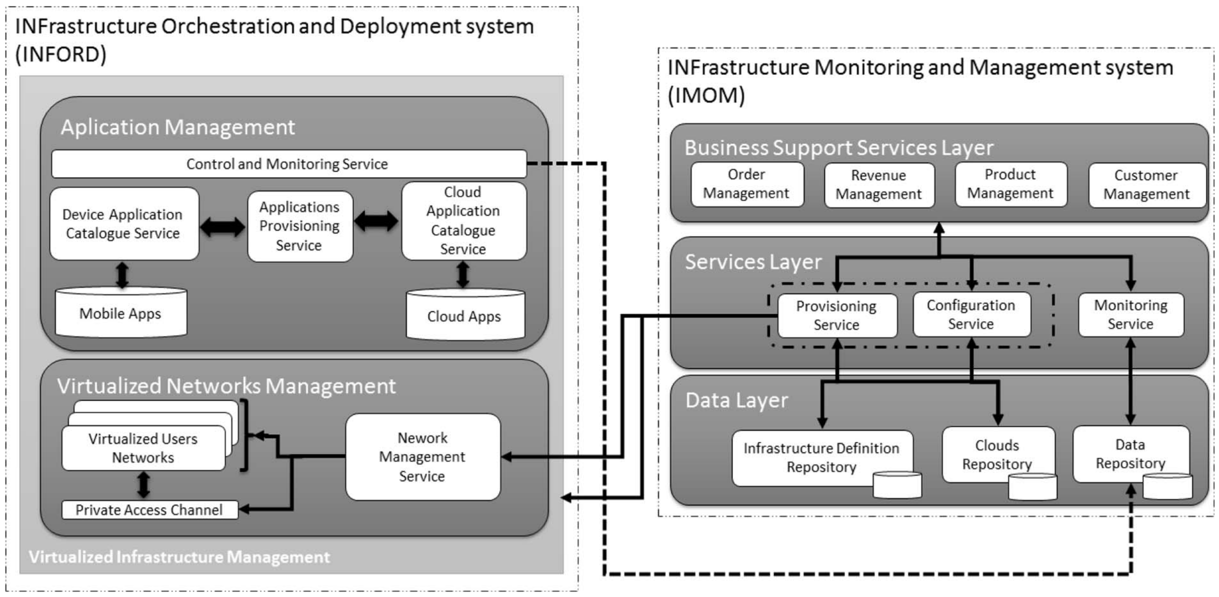
INFrastructure Orchestration and Deployment system (INFORD)
Aplication Management
Control and Monitoring Service
Device Application Catalogue Service
Applications Provisioning Service
Cloud Application Catalogue Service
Mobile Apps
Cloud Apps
Virtualized Networks Management
Virtualized Users Networks
Nework Management Service
Private Access Channel
Virtualized Infrastructure Management
INFrastructure Monitoring and Management system (IMOM)
Business Support Services Layer
Order Management
Revenue Management
Product Management
Customer Management
Services Layer
Provisioning Service
Configuration Service
Monitoring Service
Data Layer
Infrastructure Definition Repository
Clouds Repository
Data Repository

Fig. 2. This figure shows the two main blocks of architecture presented in this article and the relationship. IMOM provides mechanisms for specifying and deploying services in INFORD. INFORD provides a multiplatform software environment where to deploy services defined in IMOM.

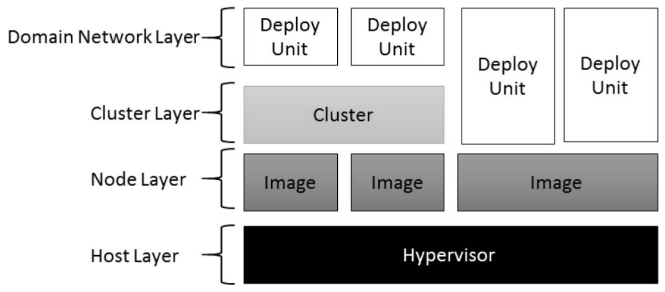

Fig. 3. Illustrates the position of the Domain Network layer, Cluster layer, Node layer, and Host layer in the stack of layers.

an operating system image which provides a standard execution environment on which CL or DNL is executed. Finally, Host layer (HL) has the capacity to support one or more Node layers. It is composed of a hypervisor that allows the management of the life cycle of NL. To manage the stack of software layers described there are several initiatives such as TOSCA[26] or NetConf/Yang [27] and [28] where the operator would define a set of configuration files to specify the architecture and deploy it. However, those mechanisms are rigid because they use templates which have static content and are written for a configuration with a specific objective. In the case of updating that configuration, it is necessary to manually enter the changes in the template. In order to solve this problem, we have developed the Configuration Domain concept (CD) which is compatible with DNL, CL, NL and HL. Each CD is composed by 3 blocks which are: templates, configuration and interfaces blocks.

The template block has schemes to define components of a domain. Each schema defines the component to be instantiated through a template with a basic configuration. Configuration block contains settings which must be applied to each of the components instantiated through templates. The interface block specifies the access points that allows to connect a component with other components instantiated. The composition of layers in CD facilitates to manage software by levels. It allows to work with an incremental configuration model where elements are instantiated from the lowest level (such as virtual machines, containers, etc.) to the highest levels (applications or services). Our solution implements this mechanism and SPs are able to deploy complex systems over INFORD. To achieve this, the SP selects the services required from the Data Layer repository. Afterward, the orchestrator (composed by PS and CS) is in charge of calculating the necessary configuration to deploy those services. This requires the orchestrator to take into account the technical characteristics of the infrastructure where INFORD will be deployed. Decomposing a system in different software layers has advantages because the orchestrator is able to work with each layer in a particular way. Each layer defines its own data model and operating rules for that domain. This approach fits very well with semantic technologies able to infer actions and generate a workflow of tasks according to rules and objectives defined [29].

Fig. 4 shows the structure which defines the process of deployment and configuration of the services and software elements. In this structure, the template section shows the files that define the element to be created, the section can have several definitions for the same component but using different frameworks and tools which facilitate its deployment in multiple environments (for example frameworks). In Fig. 4, the reader can see that the element can be instantiated in microservices container systems, or it can be installed in a baremetal environment using tools to set up and manage computers. If a virtualized environment is necessary then it could use some virtualization tool to create the necessary environment before installing the software. It is possible to include more tools that specify the deployment of the software elements but the orchestrator will only use those suits. The configuration section is intended to refine settings that could not be completed by the templates section through the use of scripts. Finally,

```
---
name: Component name
description: Any description
# Templates block for software deployment
templates:
  - container management system:
    - file: container-compose.yml
    - additionalFiles: [file1 , file2]
  - software configuration tool:
    - file: templateConfiguration.yml
  - virtual environment configuration tool:
    - file: configurationFile
    - additionalFiles: [file1.sh]
# Configuration block
configurations
  - scripts:
      file: configuration.sh
  - configuration management tool:
      file:
# Interface block
interfaces:
  file: interfacesDefinition.yml
---
```

Fig. 4. Shows a definition of a software element stored in the Data Layer.

the interface block contains a specification of how other components or services have to connect to this component once it has been deployed.

The structure showed in Fig. 4 is also valid for configuring and displaying the different layers shown in Fig. 3. It is only necessary to adjust the template and the appropriate configuration for the desired layer.

The Monitoring Service (MS) measures the quality of the services offered to SC. The MS receives information from the data repository and generates statistics and reports.

Finally, the Data Layer (DL) provides three independent repositories in which different data can be persisted. The Repository Definition Infrastructure (RDI) persists information concerning software elements and services. The Repository Clouds (RC) stores the inventory of INFORD infrastructures registered and the Repository Data (RD) to store historical information about the operation of each instantiated service.

2) Infrastructure Orchestration and Deployment System: INFORD offers a virtualized and flexible environment. Its objective is to deploy a service ecosystem oriented to satisfy the requirements of SP. The architecture of INFORD and its relation with IMOM is shown in Fig. 2.

INFORD has two layers. The lower layer, called Virtualized Network Manager (VNM), is the closest to SC. VNM is responsible for providing resources to create a virtual ecosystem in the cloud composed of the representation of consumer devices and services. Consumer devices are represented by Software Virtualization Device (sVD). sVDs are abstractions that allow SC devices to interact with other components represented as part of the ecosystem. In this way, components can transparently interact between them regardless of whether they are physical or virtual devices. This virtual environment is called Virtual User Network (VUN) which contains virtual devices connected to a virtual network.

As can be seen in Fig. 5, the main component of the VUN is the Local Network Management (LNM), its function is to manage the connections between devices connected to the VUN. LNM basically consists of a switching component with a control plane governed by a Software Defined Network (SDN) stack. The advantage of integrating NFV and SDN technologies makes it possible to benefit from emerging security mechanisms such as [4] and [30].

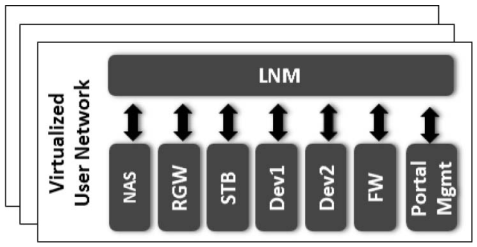

Fig. 5. The figure shows an example of VUN composed of several home devices.

The VUN is ideal for instantiating several virtual devices as firewalls, residential gateways, network attached storage and management portal among others. Other devices showed in Fig. 5 such as STB, Dev1 and Dev2, are the representation of physical devices. In general, every consumer device, such as tablets, smartphones, SmartTVs ... willing to interact with virtual services will be instantiated as an SVD.

The VUN is connected to the physical network by means of a Private Access Channel (PAC) component. The PAC is a set of SDN rules which tunnels the traffic from the user network to the VNM. If either a consumer device changes its network or a new consumer device is added to the VUN, a PAC is created.

The Application Management (AP) layer provides software components (such as virtual devices or services) to the VUN so that they are instantiated on the network and can be used by customers. To be precise, the Application Provisioning Service (APS) is responsible for providing a service catalog. APS provides to SC with a search engine of applications based on a semantic search engine adapted from [31]. The APS does not only provide software to be installed on consumer devices, but also virtual devices and services. The APS relies on two different catalogs: the Device Application Catalog service (DAC) and Cloud Application catalog service (CAC). The DAC service provides the necessary client software (if any) that should be installed in consumer devices to interact with virtual devices located in VUN. The CAC service contains components to be deployed in the VUN as storage, servers...

3) IMOM and INFORD Working Together: Our solution is the point of union that integrates different participants in the ecosystem of digital services that currently exist. Each of the participants has a specific role and interacts with the platform in a different way, see Fig. 6.

The BSS layer is the common entry point for different IMOM users. The BSS layer consists of a set of services segmented by user profiles. Each user profile is defined by the operations that can perform. Thus, operations related to the publication and maintenance of software component repositories are the responsibility of MS. The MS will connect to the platform through the BSS layer and will only be able to interact with those services used to publish and update components. In the same way, SPs will only be able to access the software components and actions allowed (to publish and

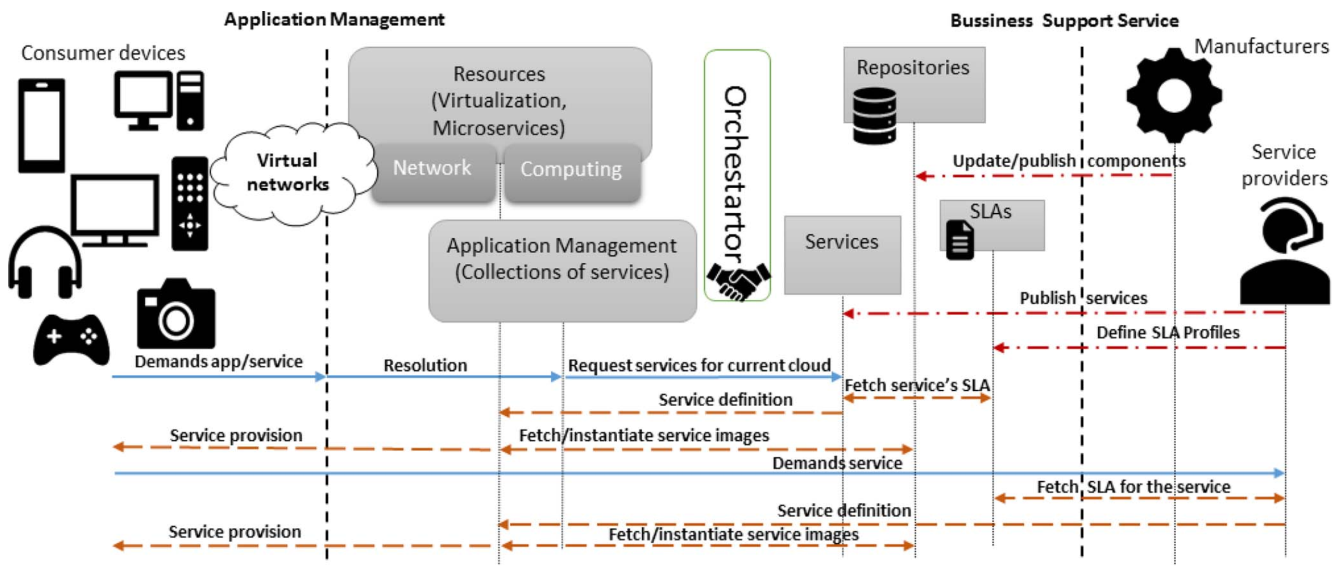

Fig. 6. Illustrates the role of the different components during application/service request and instantiation.

deploy services over infrastructures in IMOM). The publication or update of software components on the platform requires that SM provides complementary information to specifies and defines the characteristics of each component. This process is necessary for orchestration tasks and to manage the catalog of services. In the case of SPs, the process is similar for the publication and updating of services.

When a SP wants to deploy a service, it makes a request to the orchestrator indicating services to be deployed. The orchestrator requests information from the repositories about the definition of each service and technical characteristics of the INFORD platform where they will be deployed. Then, it generates the optimal configurations to carry them out.

Finally, the SCs are the other users who will connect to the platform to consult the catalog of available services and will contract those that are of interest to them.

Once the resources have been contracted, INFORD instantiates the VUN of each user from the IMOM specification received. An user can add devices/services to their VUN and those, thanks to the Application Management (AM), will automatically be instantiated and configured. In addition, a device can demand access to an application or service to fulfill its purpose. The device sends a request to the orchestrator through an application management interface (infrastructure layers). The request should be resolved to a specific description for the involved orchestrator. Thus, the orchestrator will rely on the framework to get that description and will fetch every necessary component from appropriate repositories (maintained by SMs and SPs). After that, applications or services are going to be accessed by the devices.

It is possible that a consumer device demands access to an application or service, service advertised by a SP. Then, the device requests services to the SP, the SP connects to the orchestrator which determines the computing infrastructure to be used. Then, the orchestrator deploys the service and redirects the device to the infrastructure selected or increase the available resources to cope with the demand. And the orchestrator will fetch every necessary component from the appropriate repositories (maintained by manufacturers and SPs). After that, the device will be able to access the application or service. The orchestrator before deploying/instantiating a service will query information about SLAs. Thus, the orchestrator will configure the service with the parameters defined in the SLA corresponding to the service to be deployed.

V. Implementation and Results

This section shows a prototype for evaluation with different configuration options. Several analyzes have been performed to compare the behaviour of the solution under multiple infrastructures.

A. *Target Scenario Emulation*

This section presents an architecture and the main components of the prototype developed. Figure 7 shows a diagram of the location and connection of the components. The prototype developed consists of a Smart video service (SVS) that manages users, provides the multimedia catalogue and streams video. Users interact with SVS through a graphical interface (GUI) based on responsive Web templates. When a user requests a video through the GUI, the controller component (CC) receives an http request with data in the body, those data have a json structure which contains metadata with information about the player. Then, the CC connects to the BD component to obtain stored patterns that help decide which video is most appropriate by taking into account those metadata. Next, CC retrieves the video according to its characteristics from the repository component, and finally, the video is delivered to the end-user through the Web Server component.

The SVS has been defined through configuration templates in the IMOM Data Layer. Template configuration makes it possible to deploy the SVS over different execution environments.

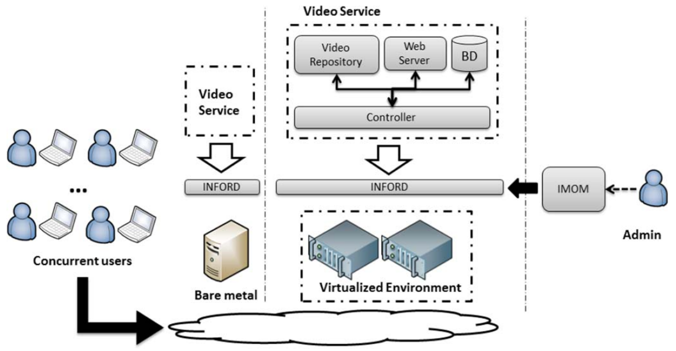

Fig. 7. Architecture for testing the proposed solution. The video service is deployed over IMOM. IMOM can be deployed over Bare metal or virtualized infrastructure. The services are consumed concurrently by users. The administrator plans the SVS deployment via IMOM and finally deploys it over INFORD.

B. Performance Analysis

Unlike other works, the proposed solution does not improve performance for services or applications that are deployed in the proposed architecture. The target is to improve the compatibility, management and deployment of applications and services. For this reason, this section presents a comparison of the operation of a service in different test environments. For this reason, we have defined three types of test environments. The first test environment (B) consists of a bare metal server, the second test environment (L) consists of an OS-level virtualization layer (such as Linux Container [32]), and finally, the third test environment (D) consists of an OS-level virtualization layer oriented to microservices (such as Docker [32]).

To analyze how the prototype works, we have used the open source benchmarking tool JMeter [33] to measure response times and connection time for each request with a different number of concurrent users. The test carried out involves the connection and viewing of multimedia content by each simulated user. The test simulates 125 concurrent users who were progressively making requests reaching their maximum load at 20 seconds. We have measured the load of the system according to the time taken to establish each connection. The goal is to analyze how the connect times vary by increasing users who access the service.

For this, we use Figure 8 where we show the connection times of each of the environments to facilitate their comparison. As you can see in the Figure 8, the measurements obtained for each of the environments are very similar. To analyze the results from a different perspective we have created figure 9 which shows the histogram of the connection times obtained for each test environment. The reader can see how the most requests are resolved before 10 msg and also the measures obtained from the different testing environments are not very different.

The B environment presents the best results with slightly shorter times than virtualized environments D and L. This result is expected because, despite the optimization of virtualization environments, the virtualization layer that encapsulates the software inside introduces a small delay in processing tasks. The L environment is slightly faster than the

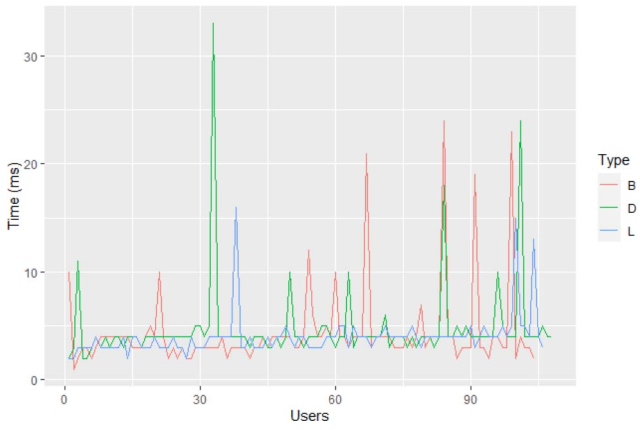

Fig. 8. Illustrates a comparison of the connection times obtained in the tests for each infrastructure depending on the number of users. Nomenclature: B is Bare metal infrastructure. L is Linux Containers infrastructure and D is microservices containers infrastructure.

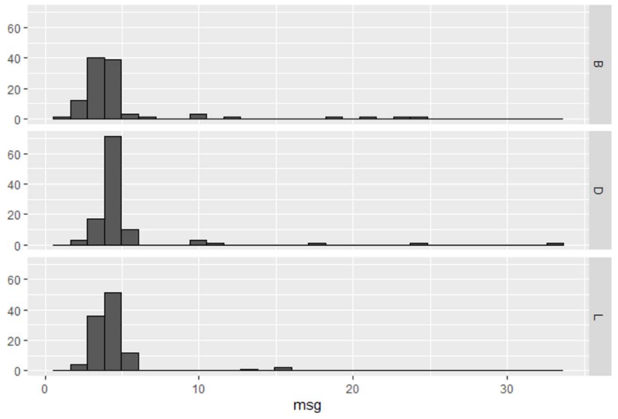

Fig. 9. Illustrates histograms of the connection times for each tested infrastructure. Nomenclature: B is Bare metal infrastructure. L is Linux Containers infrastructure and D is microservices containers infrastructure.

D environment. We think it may be due to the features of the software used for virtualization. We are aware that there are many virtualization options but the goal of this work is not to measure the performance of each of the possible options. Our objective is to verify the flexibility to be able to deploy services in different environments. And we've also been able to check the behaviour of the same service over different environments. The performance conclusions obtained by these tests may be used by software architects to decide which environment is best suited to deploy services. For this decision, assessing performance is an important factor but the characteristics and properties of each runtime environment will also need to be taken into account. Because each virtualization technology provides special properties and therefore, each of them is ideal for specific situations and requirements.

VI. Conclusion

In this work, we propose a programmable architecture capable of instantiating services and multimedia home devices in the cloud. Due to the heterogeneity of cloud infrastructures, the proposed architecture uses virtualization mechanisms to solve interoperability and lock-in problems adapting to multiple cloud environments, even edge-based infrastructures computing. However, the ability to adapt to multiple infrastructures is complex, and traditional orchestrators used to deploy software have limitations that reduce their reach to certain platforms. This work proposes to use a model segmented by software layers and also the management of each layer through templates, configurations, and interfaces blocks. This allows orchestrators to be able to deploy and easily interconnect software parts. We believe that our solution will allow home users to enjoy new digital services at a lower price and higher quality. While infrastructure management is easier for service providers, software manufacturers, and cloud providers.

References

[1] "Cisco visual networking index: Global mobile data traffic forecast update, 2016–2021," Cisco, San Jose, CA, USA, Rep., 2017.

[2] J. Opara-Martins, R. Sahandi, and F. Tian, "A business analysis of cloud computing: Data security and contract lock-in issues," in *Proc. 10th Int. Conf. P2P Parallel Grid Cloud Internet Comput. (3PGCIC)*, Nov. 2015, pp. 665–670.

[3] M. Chiosi *et al.*, "Network functions virtualisation—Introductory white paper," ETSI, Sophia Antipolis, France, Rep., 2012. [Online]. Available: http://portal.etsi.org/NFV/NFV_White_Paper.pdf

[4] S. Shirali-Shahreza and Y. Ganjali, "Protecting home user devices with an SDN-based firewall," *IEEE Trans. Consum. Electron.*, vol. 64, no. 1, pp. 92–100, Feb. 2018.

[5] J. Humble and D. Farley, *Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation* (Addison-Wesley Signature Series Fowler). London, U.K.: Pearson Educ., 2010.

[6] K. Misra, *OSS for Telecom Networks: An Introduction to Networks Management* (Computer Communications and Networks). London, U.K.: Springer, 2004.

[7] "IMT-2020 network management and orchestration framework," Int. Telecommun. Union, Geneva, Switzerland, ITU-Recommendation Y.3111, 2017. [Online]. Available: https://www.itu.int/rec/T-REC-Y.3111-201709-I/en

[8] "Telecommunications and Internet converged services and protocols for advanced networking (TISPAN); NGN management; Operations support systems architecture," ETSI, Sophia Antipolis, France, Rep. DTS/TISPAN-08007-NGN-R1, 2005. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/188000_188099/188001/01.01.01_60/ts_188001v010101p.pdf

[9] J. Quittek *et al.*, "Network functions virtualisation (NFV); Management and orchestration," ETSI, Sophia Antipolis, France, Rep. DGS/NFV-MAN001, 2014. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_NFV-MAN001v010101p.pdf

[10] Alcatel-Lucent, "Lifecycle service orchestration (LSO): Reference architecture and framework," MEF, Rep. MEF 55, 2016. [Online]. Available: https://www.mef.net/Assets/Technical_Specifications/PDF/MEF_55.pdf

[11] R. Souza, K. Dias, and S. Fernandes, "NFV data centers: A systematic review," *IEEE Access*, vol. 8, pp. 51713–51735, 2020.

[12] J. Chen, Y. Chen, S. Tsai, and Y. Lin, "Implementing NFV system with openstack," in *Proc. IEEE Conf. Dependable Secure Comput.*, 2017, pp. 188–194.

[13] M. A. Kourtis *et al.*, "T-NOVA: An open-source MANO stack for NFV infrastructures," *IEEE Trans. Netw. Service Manag.*, vol. 14, no. 3, pp. 586–602, Sep. 2017.

[14] R. Mijumbi, J. Serrat, J. L. Gorricho, S. Latre, M. Charalambides, and D. Lopez, "Management and orchestration challenges in network functions virtualization," *IEEE Commun. Mag.*, vol. 54, no. 1, pp. 98–105, Jan. 2016.

[15] P. Yasrebi, H. Bannazadeh, and A. Leon-Garcia, "Enhanced real time content delivery using VCPE and NFV service chaining," in *Proc. 12th Int. Conf. Netw. Service Manag. (CNSM)*, Oct. 2016, pp. 312–317.

[16] Z. Bronstein and E. Shraga, "NFV virtualisation of the home environment," in *Proc. IEEE 11th Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2014, pp. 899–904.

[17] T. Cruz, P. Simões, N. Reis, E. Monteiro, and F. Bastos, "An architecture for virtualized home gateways," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manag. (IM)*, May 2013, pp. 520–526.

[18] N. Herbaut, D. Negru, G. Xilouris, and Y. Chen, "Migrating to a NFV-based home gateway: Introducing a surrogate VNF approach," in *Proc. 6th Int. Conf. Netw. Future (NOF)*, Sep. 2015, pp. 1–7.

[19] P. Quinn and T. Nadeau, "Problem statement for service function chaining," Internet Eng. Task Force, Fremont, CA, USA, Rep. RFC 7498, 2015. [Online]. Available: https://tools.ietf.org/html/rfc7498

[20] "Networks funtions virtualisation (NFV); Use cases," ETSI, Sophia Antipolis, France, Rep. RGR/NFV-001ed121, 2017. [Online]. Available: https://www.etsi.org/deliver/etsi_gr/NFV/001_099/001/01.02.01_60/gr_nfv001v010201p.pdf

[21] F. Sánchez and D. Brazewell, "Tethered linux cpe for IP service delivery," in *Proc. 1st IEEE Conf. Netw. Softw. (NetSoft)*, Apr. 2015, pp. 1–9.

[22] R. Bonafiglia, S. Miano, S. Nuccio, F. Risso, and A. Sapio, "Enabling NFV services on resource-constrained CPEs," in *Proc. 5th IEEE Int. Conf. Cloud Netw. (Cloudnet)*, Oct. 2016, pp. 83–88.

[23] G. Faraci and G. Schembra, "An analytical model to design and manage a green SDN/NFV CPE node," *IEEE Trans. Netw. Service Manag.*, vol. 12, no. 3, pp. 435–450, Sep. 2015.

[24] C. K. Dominicini, G. L. Vassoler, M. R. N. Ribeiro, and M. Martinello, "VirtPhy: A fully programmable infrastructure for efficient NFV in small data centers," in *Proc. IEEE Conf. Netw. Funct. Virtualization Softw. Defined Netw. (NFV-SDN)*, Nov. 2016, pp. 81–86.

[25] "NFV 002 v1.2.1 network functions virtualisation (NFV); Architectural framework, ETSI ISG," ETSI, Sophia Antipolis, France, Rep., 2004.

[26] "Topology and orchestration specification for cloud applications version 1.0," OASIS Open, Rep. TOSCA-v1.0, Nov. 2013. [Online]. Available: http://docs.oasis-open.org/tosca/TOSCA/v1.0/os/TOSCA-v1.0-os.html

[27] R. Enns, M. Bjorklund, J. Schoenwaelder, and A. Bierman, "Network configuration protocol (NETCONF)," Internet Eng. Task Force, Fremont, CA, USA, Rep., 2011.

[28] M. Bjorklund, "YANG—A data modeling language for the network configuration protocol (NETCONF)," Internet Eng. Task Force, Fremont, CA, USA, Rep., 2010.

[29] S. I. Kim and H. S. Kim, "Semantic ontology-based NFV service modeling," in *Proc. 10th Int. Conf. Ubiquitous Future Netw. (ICUFN)*, Jul. 2018, pp. 674–678.

[30] S. Luo, J. Wu, J. Li, and L. Guo, "A multi-stage attack mitigation mechanism for software-defined home networks," *IEEE Trans. Consum. Electron.*, vol. 62, no. 2, pp. 200–207, May 2016.

[31] H. Hong and D. Lee, "A personalized refinement technique for semantic multimedia content search in smart TV," *IEEE Trans. Consum. Electron.*, vol. 61, no. 4, pp. 581–587, Nov. 2015.

[32] D. Bernstein, "Containers and cloud: From LXC to docker to kubernetes," *IEEE Cloud Comput.*, vol. 1, no. 3, pp. 81–84, Sep. 2014.

[33] R. Abbas, Z. Sultan, and S. N. Bhatti, "Comparative analysis of automated load testing tools: Apache jmeter, microsoft visual studio (TFS), loadrunner, siege," in *Proc. Int. Conf. Commun. Technol. (ComTech)*, 2017, pp. 39–44.

Iván Bernabé-Sánchez received the degree in information technology engineering from the Carlos III University of Madrid, Leganés, Spain, in 2007. From 2008 to 2014, he was a Researcher with the Telematics Engineering Department. He has participated in several European-funded projects, such as OSAMI and Spanish-funded projects, such as HAUS, Raudos, and Raudos2. He currently works with the private company in the software department. His research interests include virtualization of devices and infrastructures defined through code, cloud and edge computing architectures, and self-configuration systems mechanisms.

Daniel Díaz-Sánchez (Senior Member, IEEE) received the Telecommunication Engineering degree from the Carlos III University of Madrid, Leganés, Spain, in 2003, and the Ph.D. degree in 2008. He joined the Telematic Engineering Department, Carlos III University of Madrid in 2004, as a Researcher cooperating with Pervasive Computing Laboratory Team. He is an Associate Professor with the Telematic Engineering Department. His research interests include distributed computing, fog computing, IoT, and smart cities.

Mario Muñoz-Organero (Member, IEEE) received the M.Sc. degree in telecommunications engineering from the Polytechnic University of Catalonia, Barcelona, Spain, in 1996, and the Ph.D. degree in telecommunications engineering from the Carlos III University of Madrid, Leganés, Spain, in 2004. He is currently a Professor of Telematics Engineering with the Universidad Carlos III de Madrid. His research interests include open service creation environments for advanced mobile communication systems and convergent networks.