Hindawi
Wireless Communications and Mobile Computing
Volume 2018, Article ID 5786936, 18 pages
https://doi.org/10.1155/2018/5786936

Research Article

Orchestration Architecture for Automatic Deployment of 5G Services from Bare Metal in Mobile Edge Computing Infrastructure

Enrique Chirivella-Perez[1], **Jose M. Alcaraz Calero**[1], **Qi Wang**[1] and **Juan Gutiérrez-Aguado**[2]

[1]*School of Engineering and Computing, University of Scotland, UK*
[2]*Departament d'Informàtica, Universitat de València, Spain*

Correspondence should be addressed to Enrique Chirivella-Perez; enrique.chirivella-perez@uws.ac.uk

Received 25 May 2018; Accepted 25 October 2018; Published 22 November 2018

Academic Editor: Rüdiger C. Pryss

The progress in realizing the Fifth Generation (5G) mobile networks has been accelerated recently towards deploying 5G prototypes with increasing scale. One of the Key Performance Indicators (KPIs) in 5G deployments is the service deployment time, which should be substantially reduced from the current 90 hours to the target 90 minutes on average as defined by the 5G Public-Private Partnership (5G-PPP). To achieve this challenging KPI, highly automated and coordinated operations are required for the 5G network management. This paper addresses this challenge by designing and prototyping a novel 5G service deployment orchestration architecture that is capable of automating and coordinating a series of complicated operations across physical infrastructure, virtual infrastructure, and service layers over a distributed mobile edge computing paradigm, in an integrated manner. Empirical results demonstrate the superior performance achieved, which meets the 5G-PPP KPI even in the most challenging scenario where 5G services are installed from bare metal.

1. Introduction

The vast majority of existing tools for service orchestration are traditionally designed for physical infrastructure and assume the existence of an operating system installed in the computers of the physical infrastructure. The number of service orchestration solutions currently available is remarkably decreased when the underlying infrastructure considered is a multitenant virtual infrastructure especially in the emerging Fifth Generation (5G) mobile networks. 5G allows multiple network operators to share the same physical infrastructure [1], consequently reducing capital and operational expenditure significantly [2]. Furthermore, such a tool is almost nonexistent when looking for a service orchestration software that is able to deal with both 5G physical and virtual infrastructures in an integrated manner to control the complete life-cycle of service deployments of each layer.

This lack of solutions has led to an approach where there is a separation between the management planes for the deployment of services in physical and virtual infrastructures, respectively.

This separation is happening mainly due to the separation of business roles among physical infrastructure provider, virtual infrastructure provider, and digital service provider [3].

However, there are some scenarios where these roles are belonging to the same entity and yet there is a lack of solutions to cover these scenarios. The current approach requires employing more than one service Orchestrator to achieve the deployment of services in physical machines and virtual machines, respectively, to allow the deployment in both infrastructures. However, this separation has two main disadvantages. Firstly, it increases significantly both capital and operational costs due to the fact that two different software architectures need to be maintained in parallel. Secondly, it slows down the management planes to react to changes of the service deployment in the infrastructures, mainly due to the fact that two different management planes

need to be synchronized and coordinated in a coherent and consistent way.

Moreover, 5G is expected to be featured with new network paradigms such as the Mobile/Multiaccess Edge Computing (MEC) architecture, where different edge networks are geographically distributed to push resources closer to end users for improved quality assurance. In terms of support for dealing with the selection of the location where to deploy services, most of the existing service orchestration tools are only able to deal with physical infrastructures. Little service orchestration software has even considered multitenant virtual infrastructures. This lack of support for the selection of the location results in the limitation of current service orchestration solutions to handle distributed infrastructures where location-specific deployment is key such as in MEC infrastructures [4].

The evolution of current service Orchestrator software towards 5G compliance and automation capabilities is especially important for 5G players such as telecommunication operators, infrastructure providers, Internet service providers and digital service providers due to the expected reduction of capital and operation costs. The magnitude of this challenge has attracted the attention of the EU 5G Public-Private Partnership (5G-PPP) program, the main research initiative in Europe for developing novel 5G systems and driving 5G standardization. One of the main technical Key Performance Indicators (KPIs) defined by 5G-PPP is "the reduction of the service creation time from the current 90 hours to 90 minutes in a 5G-compliant infrastructures" [5].

5G-compliant infrastructures require the capability to deal with MEC infrastructures where geographical distribution of edge networks is a key challenge for the effective deployment of 5G services. 5G-compliant infrastructure also requires an integrated orchestration of both physical and virtual infrastructures in order to fulfill the ambitious KPI in service creation time in particular from bare-metal infrastructures, which is the most challenging scenario and the focus of this paper. These requirements and the lack of architectures, designs, and prototypes that are able to meet such requirements have been the main motivation for this research work.

The main contributions of this research work are summarized as follows:

(i) The research proposes a new service deployment architecture that is able to handle both physical and virtual infrastructures in an integrated way, which reduces the provisioning times previously existing in other tools that provisions the physical and virtual infrastructures separately. The work presented in this contribution is not only a methodology because it also has a significant effort underneath related to the implementation of the Orchestrator that has integrated the other orchestrators that work in the different management planes related to physical machines, virtual machines and service life-cycle management. Moreover, the physical infrastructure management is extended to support geographically distributed edge networks to allow offloading the processing overhead from the datacenter (the core network).

(ii) The proposed solution enables a complete control of the life-cycle of each element in each layer of the infrastructure, covering the whole life-cycle of the deployment of 5G services from bare metal, and in turn, fulfilling the ambitious service deployment time KPI imposed by the 5G-PPP program.

(iii) The proposed solution allows the management of a multilayer multitenant 5G MEC infrastructure, and thus enables multitenancy at both the physical and virtual infrastructure layers, leading to higher granularity in terms of security through isolation between tenants and layers.

(iv) Different orchestration strategies in terms of horizontal or vertical deployment of services are investigated and compared based on either a centralized multitenant cloud infrastructure or a novel distributed multitenant MEC infrastructure.

(v) The proposed system with all the above novel technical advances has been integrated and prototyped in a real infrastructure deployment. Empirical results provide insights into optimal ways to carry out 5G service orchestration over 5G MEC-compliant infrastructures.

The rest of the paper is organized as follows: Section 2 provides a review of related work where a comparative study of existing tools is conducted to highlight the main gaps in existing tools and thus motivates a solution. Section 3 presents the proposed MEC-compliant architecture with ETSI MANO and emphasizes the new elements that the proposed solution introduces to fill the gaps identified in Section 2. Section 4 presents the detailed steps in automatic orchestrating the deployment of 5G services from bare metal. Empirical performance evaluation and analysis of different orchestration strategies is provided in Section 5, based on a realistic implementation of the proposed solution in a 5G MEC testbed. Finally, concluding remarks are presented in Section 6.

2. Related Work

Despite the fact that numerous tools are already available in the market to deploy software in an automatic way, this section will explain why the existing tools are not sufficient to meet the 5G service deployment requirements and address the KPI challenge in 5G and thus justify the contributions of this paper based on a critical review.

2.1. Requirements for 5G Service Deployment Orchestration. The 5G mobile edge computing paradigm [4] requires an Orchestrator that is able to control a large number of distributed machines that should be ready to deploy 5G services in less than 90 minutes. To this end, a set of specific requirements have been identified for the solution and are explained in the following paragraphs:

Resource Life-Cycle Control. To be able to have a complete control over physical resources and virtual resources. The solution needs to control life-cycle of each physical and virtual resource and its different states such as power on, power off, soft restart, soft shutdown, hard restart, and hard shutdown for the physical and virtual resources.

Service Life-Cycle Control. To be able to have a complete control over the services deployed in physical and virtual resources. The solution needs to control life-cycle of each service, including start, stop, configure, reconfigure, deploy, and redeploy services.

Operating System Provisioning. To be able to install a complete operating system (OS) within the physical machine. In essence, to consider the Operating System as a special service that needs to be deployed, redeployed, configured, and reconfigured. It imposes a significant number of requirements in the provisioning architecture since it needs to deal with bare-metal management.

Multitenancy Support. This refers to the mode of operation of the solution where resources are isolated to allow the sharing of them across multiple network operators (tenants). The services of each tenant are logically isolated over the same physical infrastructure. In 5G, for example, multiple network operators (tenants) can share the same physical infrastructure whilst operating independently logically.

Multizone Support. This is the capability to enable distributed deployment of services across multiple zones and to deploy services faster than in a centralized architecture by making use of data locality to speed up deployments.

Service Location Awareness. This is the capability to use the geographical position to define zones within the infrastructure. The solution should be able to deploy services in a specific location. This feature provides to the infrastructure administrator the functionality to deploy services in each zone and ensures that architecturally it is the best solution for the user.

Workflow Dependencies Resolution. This is the ability to determine if a service depends or not on another service and resolve the dependencies for the workflow of service deployment orchestration. It is noted that 5G services typically depend on various virtual and/or physical resources.

Parallel Deployment. This is the capability of the solution to support parallel deployment and to deploy new services, applications, or infrastructures. These activities may include physical or virtual procurements, configuration, tuning, staging, installation, and interoperability testing.

2.2. Comparison of Orchestration Tools. Based on the above requirements, a comparative review is carried out to analyze available tools in terms of which features they can provide to the user to meet any of the identified corresponding requirements. There are a number of existing tools for automatic deployment services as listed and outlined below:

Puppet [6] models the entire infrastructures as code in order to significantly reduce the complexity of deploying and managing distributed infrastructure services and applications. Chef [7] automates the process of managing configurations, ensuring that every node in the infrastructure is configured correctly and consistently using 'recipes'. Capistrano [8] is the extension of Chef to allow parallel deployment and interdependency resolution along the deployment of the different services. CFEngine3 [9–11] is an Orchestrator that treats each computer as an autonomous entity, obtaining its script and a few occasional pieces of information from the policy server (conductor). Juju [12] is an open-source orchestration service that provides easy scaling, automatic provisioning on a variety of providers including bare-metal, Linux containers and various cloud stacks, and a community ecosystem of best-practice service definitions. The automation of the services is created using the so-called charms. Ansible [13] is a versatile orchestration engine that can automate deployment of systems and services. Instead of a custom scripting language or code, it is very simple and shell-based. It is also agent-less, and allows a user to utilize it using SSH connectivity. Docker [14] creates native LXC sandboxes virtual containers that act like isolated scenario where the final user can deploy their own services without affecting anything else with a configuration file that automates the process installation. SaltStack [15] is a powerful and different approach to infrastructure management, by focusing on high-speed communications between large number of systems, and can perform orchestration and remote code execution. CloudFormation [16] is an AWS Orchestrator that allows spinning up any services in AWS with a predefined set of parameters. With CloudFormation, the service of resource creation is self-documenting and changes to the resources can be tracked with the help of code repository, allowing debugging. TerraForm [17] realizes multicloud deployments, which can be very challenging as many existing tools for infrastructure management are cloud-specific. Terraform is cloud-agnostic and allows a single configuration to be used to manage multiple providers, and to even handle cross-cloud dependencies. This simplifies management and orchestration, helping operators to build large-scale multi-cloud infrastructures. Scalr [18] is an on-line solution that orchestrates the deployment of services in different public cloud providers using preconfigured configurations. Heat is an OpenStack orchestration component in charge of creating a human and machine-accessible service for managing the entire life cycle of infrastructure and services. OpenMano [19] delivers an open-source management and orchestration (MANO) stack aligned with ETSI NFV Information Models. With the support from an operator-led community, it offers a production-quality orchestration stack that meets the requirements of commercial NFV networks. Cloudify [20] is an open-source cloud orchestration platform, designed to automate the deployment, configuration and reconfiguration of applications and network services across hybrid cloud and stack environments. ZOOM [21] is the implementation of the zero-touch orchestration, operations and management project to develop best practices to support both the technology and business transformation brought about by the introduction of Network Function Virtualization (NFV) and Software-Defined Networking (SDN). OPNFV [22] project seeks to provide orchestration functionalities for testing scenarios. Most of the time, they are emulating/simulating those functionalities executing different procedures and/or

requests in parallel to different components. OpenBaton [23] is the result of an agile design process having as major objective the development of an extensible and customizable framework capable of orchestrating network services across heterogeneous NFV Infrastructures.

However, none of the above tools are able to meet all the requirements identified. For instance, all of them have the crucial limitation of not being able to control the life cycle in all the layers (i.e., physical resources, virtual resources and service) of the architecture.

Table 1 shows a comparison in terms of the features of the most popular tools currently available. This comparison provides a comparative overview of these tools in terms of capabilities that are needed to meet the requirements for 5G service deployment orchestration. In addition, Table 1 indicates an estimation of the cost reduction by the tools to different degrees: High (H), Medium (M), or Low (L). Most of these tools in Table 1 are focused mainly on one or two of the three layers in a 5G MEC paradigm. The main motivation of this paper is to provide a complete solution that enables a 5G infrastructure administrator to achieve a complete control of the life-cycle of each element in each layer of the infrastructure, thereby allowing the automatic deployment of 5G services. The last entry of the table highlights the comprehensive contributions provided in this paper.

2.3. Other Related Initiatives and Work. In addition to the existing tools analyzed above, there are a number of related ongoing standardization activities and industry initiatives.

Firstly, the Topology and Orchestration Specification for Cloud Applications (TOSCA) [24] unifies invocation of scripts and services in a generic way. Under this umbrella, the prototype approach proposed in [25] realizes TOSCA based on standards-based manner. Caballer et al. [26] propose an orchestration model that can be used with open-source software like OpenStack or hybrid infrastructures and leverages TOSCA as a standard modelling language. Zimmermann et al. [27] propose a tool to automate the provisioning and do the integration of an Analytic tool to process production steps, overall of the manufacturing processes in industrial Environments using TOSCA.

Secondly, the ETSI Zero-touch Network and Service Management (ZSM) [28] Industry Specification Group was recently (December 2017) launched with the main objective to gain benefits of automatic network and services operation in 5G. ZSM highlights the usefulness of standardization work in this area. Since it is a new initiative, no significant work has been reported. Nevertheless, this latest initiative further emphasizes the timeliness and necessity to research and develop novel architectures and tools, e.g., for automatic service orchestration, which is the focused contribution of the proposed work in this paper.

Moreover, some additional alternative approaches have been proposed. For example, Vukojevic et al. [29] introduce a provisioning middleware that optimizes the standard provisioning and deprovisioning behaviour and thus improves cost and time efficiency. Demchenko et al. [30] propose Zero-Touch Provisioning to provide access to education and research and support new collaborative applications that are becoming increasingly complex and dynamic in their scale, enabling using distributed resources that require advanced networking services. Karakostas [12] proposes an autonomic cloud configuration and deployment environment that allows automatic deployment, configuration and reconfiguration in a cloud configuration. The solution only works in legacy cloud computing architectures being able to install services in virtual environments. Kumar et al. [31] propose automated provisioning of applications in the cloud using Amazon as an IaaS provider and Ansible as the orchestration engine to automate the deployment. Demchenko et al. [32] discuss the importance of automation tools for deployment and management applications for Big Data on the SlipStream cloud automation platform.

The two most closely related tools in terms of capabilities compared to our proposed solution are Juju and Terraform, with Juju being the closest one. In fact, our proposal is based on Juju software to provide the orchestration capabilities in order to deploy and control the life cycle of physical and virtual services. Our proposal can be considered as an extension to Juju to support mobile edge computing infrastructure with multiple geographical zones and to support parallel deployments of work flow dependencies in physical and virtual resources allocation.

The above initiatives and technical advances contribute to achieving more automatic network and service management. However, a solution that fulfills the listed requirements for 5G service deployment orchestration is still largely missing and it is the main motivation of this research work.

3. MEC Infrastructure for the Automatic Deployment of 5G Services from Bare Metal

To address the gaps identified in the related work, a novel 5G MEC service deployment orchestration architecture is proposed for provisioning services over a 5G MEC infrastructure that is geographically distributed across different locations, as shown in Figure 1. The proposed architecture can be deployed in two complementary ways. One way is a conventional centralized deployment where the architecture relies on only one Controller (see the Datacenter Management Zone in Figure 1) to carry out the provisioning of services across all the zones of the infrastructure. This is the current design for multitenant cloud infrastructures. The other way is a more modern distributed scalable deployment where the architecture relies on Edge controllers available in each of the geographical zones of the infrastructure. This approach is better compliant with the definition of the 5G MEC architecture and helps the scalability requirements associated with 5G Infrastructures.

Figure 1 is divided into two parts. The right side of it contains all the architectural elements in charge of the management and orchestration plane of the infrastructure. There is also in the middle of the Figure the networking part providing the interconnectivity infrastructure. This is a logical diagram and the networks presented can be deployed over the same physical port depending on the deployment strategy in production-grade deployments. Regarding interconnectivity, the Management Infrastructure Network

Table 1: Comparison of orchestration tools.

		Juju	Chef	puppet	Ansible	SaltStack	Cloud-Formation	Terraform	CF3Engine	Capistrano	Scalr	Heat	Cloudify	OpenSource-Mano	**Our Contribution**
OS	OS Provisioning	√	✗	✗	✗	✗	✗	√	✗	✗	✗	✗	✗	✗	√
Life Cycle	Physical Resource	√	✗	√	√	✗	✗	√	✗	✗	✗	✗	√	✗	√
	Virtual Resource	√	✗	✗	√	✗	√	√	✗	✗	✗	✗	√	√	√
	Service	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Multi Tenancy	Physical Resource	√	✗	✗	✗	✗	✗	√	✗	✗	✗	✗	✗	✗	√
	Virtual Resource	√	√	√	✗	✗	✗	√	✗	√	✗	√	√	√	√
Multi Zone	Physical Resource	√	✗	√	√	✗	√	✗	✗	✗	√	✗	√	✗	√
	Virtual Resource	✗	√	√	√	✗	✗	✗	✗	√	√	√	√	✗	√
	Physical <-> Virtual	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	√
Service Location	Zone	√	√	√	√	✗	√	√	√	√	✗	√	✗	✗	√
	Physical Resource	√	√	√	√	√	✗	√	√	√	√	✗	√	✗	√
	Virtual Resource	√	√	√	√	√	√	√	√	√	√	√	√	√	√
WorkFlow Dependencies	Physical Resource	√	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	√
	Virtual Resource	√	√	√	✗	✗	✗	✗	✗	√	✗	✗	✗	√	√
	Service	√	√	√	√	√	√	√	√	√	√	✗	√	√	√
Parallel Deployment	Physical Resource	✗	√	√	√	✗	✗	✗	✗	√	✗	✗	✗	✗	√
	Virtual Resource	✗	√	√	√	√	✗	✗	✗	√	✗	✗	✗	√	√
	Service	√	√	√	√	√	√	√	✗	√	✗	✗	√	√	√
-	Cost Reduction	H	L	M	M	L	M	M	L	M	M	L	L	L	H
Associated Publications	References	[42–44]	[45–47]	[48–50]	[51–53]	[54, 55]	[56–58]	[59–61]	[62–64]	[65–67]	[68–70]	[71–73]	[26, 74, 75]	[76–78]	

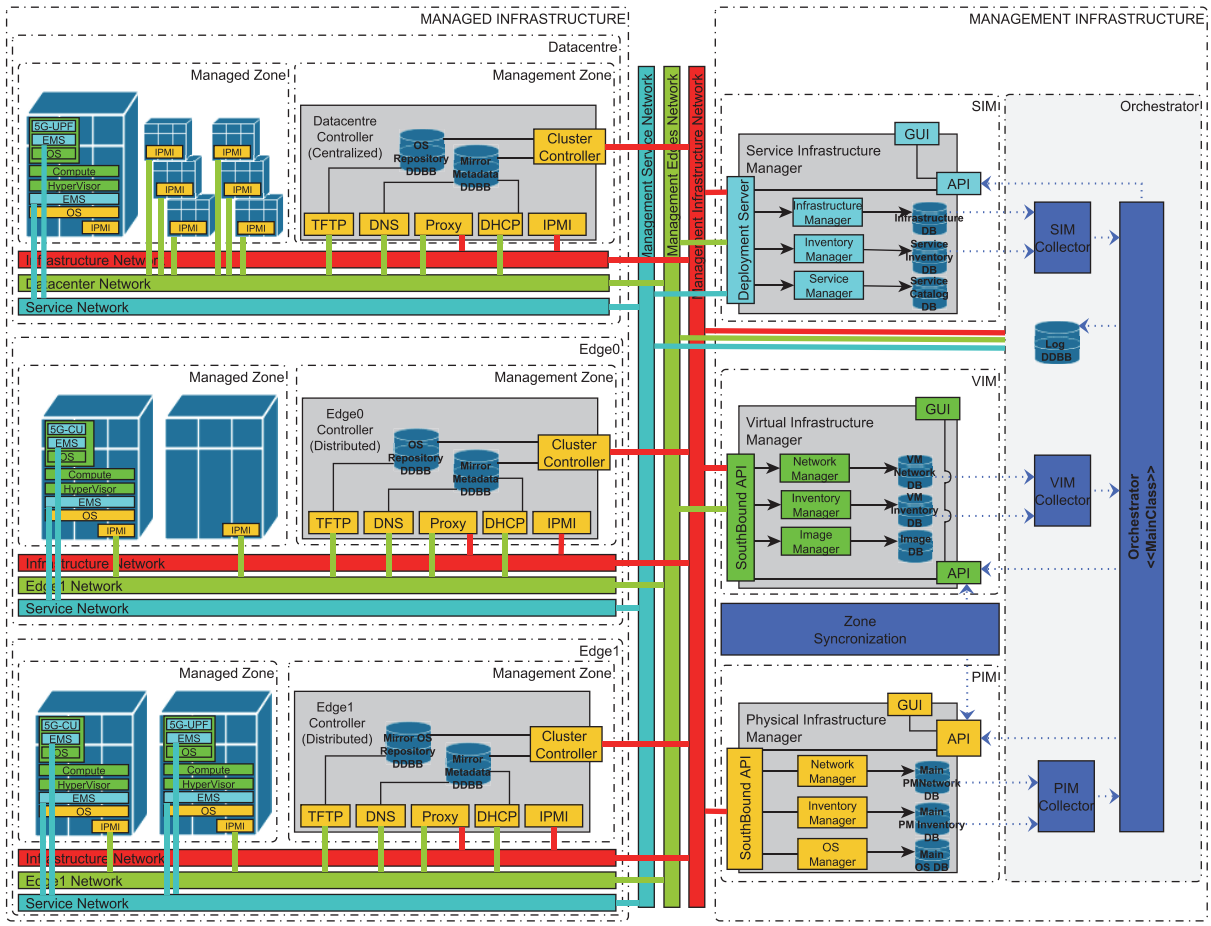

FIGURE 1: Overview of the proposed 5G MEC service deployment orchestration architecture.

allows infrastructure administrators to provision all services deployed over the physical resources of the infrastructure, including an operating system. It expands across all geographical locations. The Management Edges Network allows connectivity between all the physical resources deployed over the same geographical location for localized management. Figure 1 shows the interconnection between different Edges Networks, although the visibility is filtered by the usage of different Virtual LANs (VLANs) [33] according to the location. The Management Service Network allows the Service Infrastructure Manager to deploy services in all places of the infrastructure and represents the data path for the 5G services.

The right part of Figure 1 is the management plane and is composed of three horizontal elements and one vertical element. The horizontal elements are the Physical Infrastructure Manager (PIM), Virtual Infrastructure Manager (VIM) and Service Infrastructure Manager (SIM). These elements are explained in more detail below:

Physical Infrastructure Manager (PIM). PIM controls physical resources and the life-cycle of all the hardware in the infrastructure. To achieve so, it maintains and manages a catalog of the different operating systems to be provisioned, and also maintains an inventory of the hardware resources, the operating systems deployed in each hardware resource, the life-cycle status of such hardware resources and the storage and network configuration. This information is stored in the three databases shown in the PIM in Figure 1, namely OS DB, PM Inventory and PM Configuration, respectively. PIM has a GUI and an API to allow both Users and commands to manage the hardware resources of the physical infrastructure. The Application Programming Interface (API) allows access via REST or Web Socket, and this Northbound interface allows the Orchestrator to interact with the PIM. The PIM implementation used in our testbed is based on Metal-as-a-Service [34] provided by Ubuntu.

There is an Edge Controller node in each of the locations in charge of performing the management of the hardware resources of this zone to provide scalability. PIM is connected to the Management Infrastructure Network allowing databases in the Edge Controller to synchronize different geographical locations.

Edge Controller provides an operating system to all hardware elements in the zone. This operating system contains different types of configurations depending on the element managed such as servers, switches, routers and customized configuration for network interfaces, hard drive layout, certificates, users, and policies of each element. Each Edge Controller contains two databases that are mirrored from the databases in the Physical Infrastructure Manager.

Each controller has its own servers for Dynamic Host Configuration Protocol (DHCP), Domain Name System (DNS) [35], Trivial File Transfer Protocol (TFTP) [36] and Proxy. Each zone contains its own isolated Network used to provision the operating system and to manage the compute node through its Intelligent Power Management Interface (IPMI) [37].

Virtual Infrastructure Manager (VIM). VIM controls virtual resources and the life-cycle of all the virtual hardware resources in the virtual infrastructure. VIM allows sharing the physical resources over more than one tenant (i.e., multitenancy) by dealing with virtualization technologies. Among others, it is in charge of keeping a catalog of Virtual Machine (VM) images, an inventory of the existing VMs, an inventory of the virtual networks created for each of the tenants. That information is stored in the three DBs shown in the VIM depicted in Figure 1. VIM also takes the role to determine where to deploy the VMs and to perform the deployment in the indicated zone.

VIM exchanges messages with the compute services using a message bus architecture so that each of the compute services running in each of the hardware computers allow the management of such hardware from different geographical locations. VIM allows controlling the life-cycle of each virtual resource in the infrastructure such as start, stop, soft shutdown, hard shutdown, soft reboot, hard reboot. VIM exposes a GUI for the user and a northbound API that supports the control of all virtual elements in the infrastructure. The VIM implementation used in our testbed is based on OpenStack [38].

Service Infrastructure Manager (SIM). SIM controls services deployed over both virtual and physical resources and the life-cycle of all the virtual and physical resources in the infrastructure. It is responsible to deploy services over the Physical and Virtual Layers, controlling their life-cycle, allowing deploy, redeploy, start, stop, reconfigure, and upgrade. SIM maintains 3 DBs with the inventory of the deployed services, the catalog of managed resources and the different endpoints to interacts with the VIM and PIM in order to request and free resources. SIM has a GUI and API that allows deploying the services in every place of the infrastructure. The SIM implementation used in our testbed is based on Ubuntu Juju [12]. This SIM is referred to as VNFM in ETSI MANO architectures. In our proposed architecture, the SIM is extended from VNFM to incorporate additional functionality not only for the deployment over virtual resources but also over physical ones.

Orchestrator. The Orchestrator interacts with each of the three management components described (PIM, VIM and SIM) in order to orchestrate the automatic deployment and control of services and resources over the 5G MEC infrastructure. To this end, the Orchestrator interacts with the APIs exposed by SIM, VIM and PIM in order to orchestrate the steps involved in the deployment of resources and services. However, in order to determine such a set of steps, the Orchestrator should be aware of the status of the resources and services in real-time to calculate the steps required to achieve the desired deployment of services. For this purpose, the PIM Collector, VIM Controller and SIM Collector plug-ins of the Orchestrator are designed and prototyped in order to extract the information from the different DBs of the PIM, VIM and SIM, respectively, so that the Orchestrator is aware of the current status and the changes produced by every of the steps orchestrated in this status over the entire MEC architecture.

Zone Synchronization. This component is required due to the dynamic nature of the deployment of services from bare-metal. The PIM, at the provisioning phase of the operating system, is aware of the location of the new computer being commissioned. After that, the deployment of the VIM Compute services in the computer is carried out by SIM to allow VIM to take control over the virtual resources of this computer. At this moment, VIM detects the new computer but there is no way to synchronize the geographical location of the physical machine available in the PIM with the geographical location of the physical machine registered in the VIM. This is exactly the main role of this new architectural component in order to allow the management of virtual resources taking into account their different geographical locations.

The left side of Figure 1 represents the different geographical zones of the infrastructure. There are a datacenter location and a number of edges for simplicity. They are managed similarly so that it is just a naming convention to fit the MEC terminology. In a traditional cloud infrastructure, there is only 1 Management Zone whereas in MEC there are several including the edge of the network. The reader can see some deployment examples of the 5G services in the managed computers. The layout of services for such computers has an operating system together with the Element Managed Service (EMS) to allow SIM to take control of the deployment of services over the physical machine. EMS will be used to perform the deployment of a hypervisor and a VIM Computer service to allow the VIM to take control of the resources. Then, SIM will also interface with the VIM in order to perform the installation of VNFs in the edge of the network so that the VNF has installed the operating system inside together with the EMS daemon to allow SIM to also take control of the virtual resources and the deployed 5G services. In the example, the reader can see how a 5G User Packet Forward (UPF) is deployed in the datacenter location whereas a 5G Centralized Unit (CU) is deployed in an edge of the network. A detailed description of all the 5G architectural elements is provided in [39].

To clarify our architectural contribution compared with the state of the art of the different components presented in this section in order to make the orchestration architecture compliant with MEC infrastructures, it is worth enumerating our main architectural contributions as follows:

(i) PIM, based on MaaS, is integrated with the architecture to extend the traditional ETSI NFV standardized architecture and thus extend the coverage of the life-cycle to bare-metal deployments, which is a significant extension beyond the state of the art.

(ii) The orchestration component has been designed, prototyped and validated. In fact, the following section will provide a detailed explanation of all the steps carried out by the Orchestrator in order to perform

the automatic deployment of the services in MEC infrastructures.

(iii) PIM, VIM and SIM Collectors are designed, prototyped and validated to allow the interactions with MaaS, OpenStack and Juju, respectively. It is noted that the Orchestrator follows a TOSCA-compliant pluggable architecture and thus a concrete implementation is prototyped in order to validate the architecture proposed herein.

(iv) The proposed architecture resolves the problem related to the lack of synchronization between PIM and VIM with respect to geographical zone. To the best of the authors' knowledge, it is the first time to achieve this integration between PIM and VIM, which is, in fact, a clear requirement to allow an Orchestrator with true support for orchestrating services in MEC infrastructures. Thus, this gap is identified and the Zone Synchronization component is designed, prototyped and validated in order to fill this gap.

(v) Another contribution with respect to the state of the art is related to the concrete implementation used to validate the proposed infrastructure. MaaS v1.9 was adopted as the basis of PIM. This MaaS software, however, has a strong requirement for an Internet connection between the edges and the datacenter in order to allow the management plane to work. Nevertheless, in a production-grade deployment, it is very unlikely to expose the management plane of the infrastructure to the Internet. Thus, the MaaS software has been extended in order to overcome this limitation and thus be able to work with no Internet access on the management plane.

Based on the above specific architectural contributions, together with those listed in Introduction, this research is able to provide an orchestration architecture with true support for MEC architecture, to provide empirical results over service provisioning times in MEC architectures and to provide empirical results over service provisioning times of VNFs from bare-metal.

4. Orchestration of 5G Service Deployment from Bare-Metal

Figure 2 shows a detailed sequence diagram to explain the comprehensive steps involved in the orchestration process in order to achieve the deployment of a 5G VNF service in a computer located in a remote edge of the network that does not have any operating system or information inside and is just bare metal. It means that the computer should be provisioned with the operating system, the hypervisor and VIM management support and also with the 5G VNF service deployed in a given tenant network of the shared infrastructure. This workflow pushes the boundaries of the state of the art in service provisioning due to all the requirements involved in the orchestration process. Based on the literature review conducted, this type of sequences is not supported by any of the existing orchestrating tools.

The top of Figure 2 shows the main architectural components involved in the orchestration process and the Management API involved in the orchestration process (PIM, VIM and SIM). Vertically on the left side of the diagram, it shows the different states associated with both resources and services and their changes along the whole orchestration process to achieve the deployment of a 5G VNF from bare-metal in the edge of the network. All the invocation lines available in the diagram are labeled with a number that is mentioned in the text to make it easier to follow the orchestration workflow. The workflow represents the deployment of a new 5G CU VNF into a computer located in Edge1 of the network.

As a summary, there are three mayor phases involved in the automation process. The first one is related to the installation of the operating system in the physical machine. The second phases is related to the installation of the Virtual Machine operating system. And finally, the third one is related to the installation of the services inside of the VM.

The description of the steps is grouped into different subsections to make the process more readable. It is noted that the different subsections follow a logical execution flow of different phases as explained below.

4.1. Discovery Phase. The workflow starts when *Infrastructure Administrator* sends to *Orchestrator* a request to deploy a new CU VNF in Edge1 **(1)**. In order to make sure the selected machine is provisioned from scratch and no other machine is selected instead, it is assumed that there are no other machines provisioned in Edge1. However, it is also assumed that the edge location has at least 1 management node up and running where the Cluster Controller of the PIM has already been deployed and installed. The automation of this process has also been achieved. **(2)** *New Edge Machine* is powered on by the *Infrastructure Administrator*. Then, **(3)** *New Edge Machine* that was previously wired and powered on is now detected by the *MaaS Edge controller* of that zone, which is always synchronized with *MaaS Datacenter controller*. Then, **(4)** MaaS Edge requests credentials (user and password) to configure the access to the Intelligent Platform Management Interface (IPMI) [37] interface of *New Edge Machine* and thus gain control over it. In order to make the process fully automated, these credentials are extracted from a configuration file and are the user/password provided by the vendor.

(5) MaaS then overrides the configuration of the credentials of IPMI via the Baseboard Management Controller (BMC) to allow MaaS to be able to control the life-cycle of the physical machine.

The elapsed time for steps 1-5 is defined as the time between the moment when a physical machine is physically connected to the infrastructure and when it is registered in MaaS, henceforth referred to as **Discovery Time**.

4.2. Commissioning Phase. At this stage, **(6)** *New Edge Machine* is properly configured in MaaS and then *Orchestrator* is aware of the new status of the machine pulling the PIM Collector. In a separate thread, the Zone Synchronization Service (ZSS) is periodically pooling MaaS DB **(7)** so that at

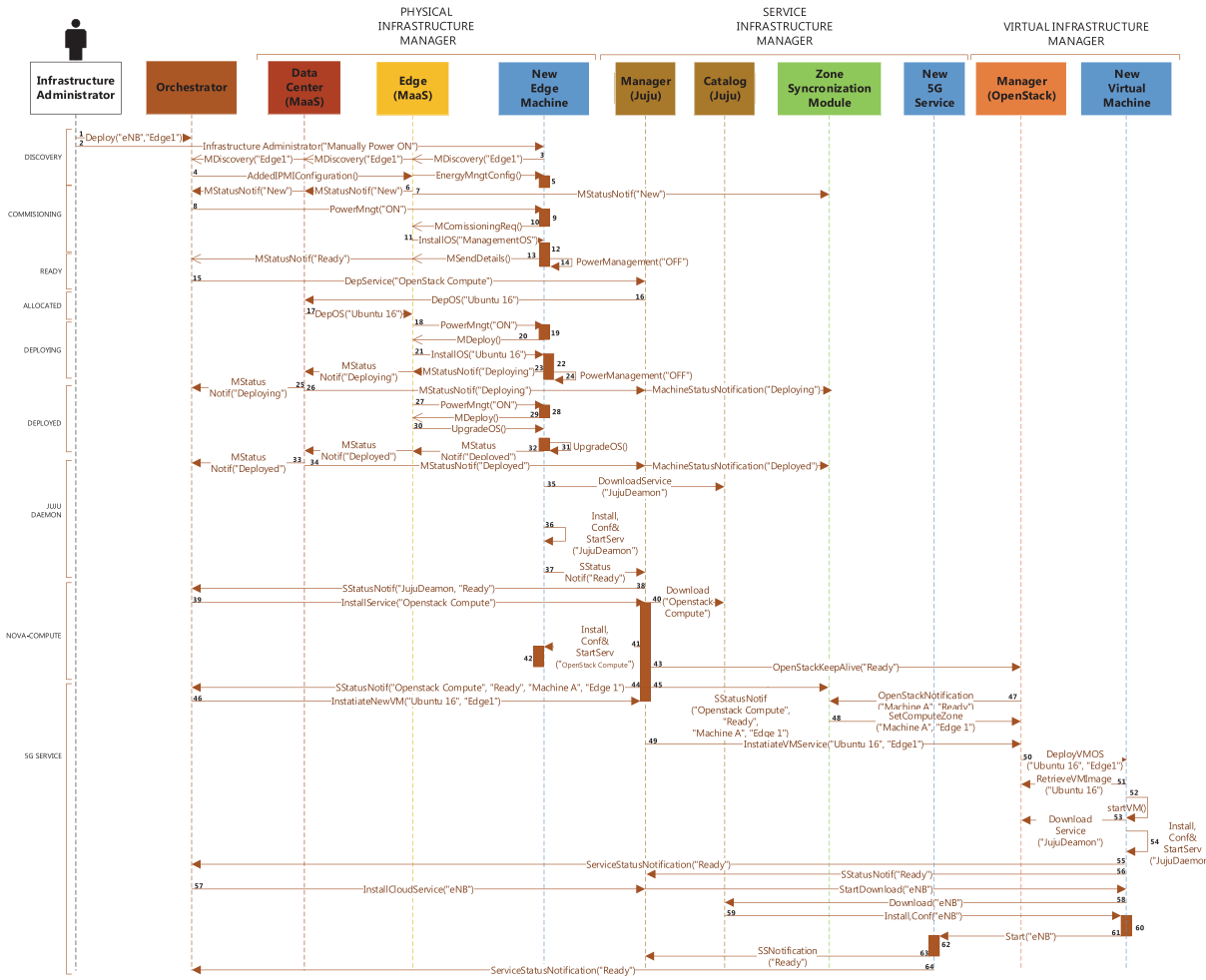

FIGURE 2: Sequence diagram for the interactions among the different architectural components available in the proposed orchestration architecture.

this stage ZSS is notified of the existence of a new machine in the zone. Then, **(8)** *Orchestrator* powers on the new physical machine detected in the edge via IPMI. **(9)** *New Edge Machine* starts the boot process using the default boot sequence in the BIOS, which is by default the network stack booting process using the PXE protocol [40].

(10) The *New Edge Machine* broadcasts in the network to discover a DHCP server that assigns dynamically an IP address and also provides the image to be retrieved from the TFTP server with the Management OS. Then, **(11)** the MaaS TFTP server located in this edge of the network returns to the *New Edge Machine* the Management OS (a minimal version of Ubuntu kernel and some utilities) and unattended installation is carried out. **(12)** The Management OS is installed, which enables MaaS to acquire the complete control over a physical machine. Moreover, as part of the installation process, the capabilities of the *New Edge Machine* are detected by MaaS regarding RAM, CPU, NICs, Storage, hardware accelerators, etc. This is exactly the main purpose of this commissioning phase.

The elapsed time for steps 6-12 is defined as the time between the moment when a physical machine is detected in MaaS and when MaaS takes full control of this machine, henceforth referred to as **Commissioning Time**.

4.3. Machine Request Phase. **(13)** When all the information required has been gathered from the *New Edge Machine*, this information is registered in *MaaS Edge*. The *New Edge Machine* then automatically powers off itself **(14)** so that the *New Edge Machine* has been commissioned properly and is ready to use. **(15)** The *Orchestrator* collects the data from the PIM Collector and requests *Juju Manager* to deploy a new OpenStack Nova-Compute service over the *New Edge physical* machine.

The elapsed time for steps 13-15 is defined as the time between the moment when MaaS takes full control of a machine and when the machine is requested as a resource where to deploy a service, henceforth referred to as **Machine Request Time**.

4.4. Allocation Phase. **(16)** *Juju Manager* requests *MaaS Data Center* to deploy a Ubuntu 16.04 in the *New Edge* Machine. **(17)** At this moment, *MaaS Data Center* allocates the *New*

Edge Machine to the owner of the administrative domain, who is the only one that is able to manage this New Edge Machine.

The elapsed time for steps 16-17 is defined as the time between the moment when the machine is requested as a resource where to deploy a service and when it is allocated to a given administrative domain (user), henceforth referred to as **Allocation Time**.

4.5. Deployment Phase. After the allocation is conducted, *MaaS Data Center* requests *MaaS Edge* to deploy a Ubuntu 16.04 in the *New Edge Machine*. **(18)** *MaaS Edge* gathers the IPMI information and powers on the *New Edge Machine*. **(19)** The *New Edge Machine* starts booting with PXE, **(20)** requests a Ubuntu 16.04 to be installed by the *MaaS Edge* together with all the configurations about network cards, hard drive layout, certificates, users, custom software to be installed and the scripts of the postinstallation, etc. **(21)** TFTP in *MaaS Edge* provides to the *New Edge Machine* the Operating System requested. **(22)** The unattended installation of Ubuntu 16.04 starts. **(23)** When the installation is successfully completed, the scripts that were previously transferred to the *New Edge Machine* are properly executed. One of these scripts is always executed for management purposes, notifying *MaaS Data Edge* that the machine has been installed properly. *MaaS Edge* synchronizes this new state of the physical resource with *MaaS Data Center*. **(24)** *New Edge Machine* reboots itself to complete all the scripts and postinstallation processes. **(25)** *MaaS Data Center* pulls the change to PIM Collector and *Orchestrator* collects the new state from there.

The elapsed time for steps 18-25 is defined as the time between the moment when the allocation of a machine to a given administrative domain (user) and when the operating system has been completely deployed and the machine is ready to be used, henceforth referred to as **Deployment Time**.

4.6. Upgrade Phase. **(26)** *MaaS Data Center* notifies *Juju Manager* which has been waiting for this interdependency to be sorted out from step 15, that the deployment is conducted. However, it is not enough and *Juju Manager* is not able to continue with the workflow until the upgrade of the operating system is carried out. After the deployment phase, by default, MaaS will perform an upgrade of the operating system to make sure that it is always up to date with no major security vulnerability discovered.

Then, **(27)** after a complete reboot of *New Edge Machine*, *MaaS Edge* powers on, via IPMI, *New Edge Machine* to finalize the installation. **(28)** *New Edge Machine* starts booting with PXE, **(29)** *New Edge Machine* requests, via PXE, the next step **(30)** and *MaaS Edge* replies now to boot from the local hard disk and performs an upgrade the operating system. **(31)** The operating system checks all the available packages in the repository and starts to be upgraded. **(32)** Once the upgrade has been done, *New Edge Machine* notifies *MaaS Datacenter* through *MaaS Edge* of the new state for *New Edge Machine*. **(33)** *MaaS Datacenter* notifies the status of *New Edge Machine* and PIM Collector collects the information and pushes the new state through *Orchestrator*. **(34)** *MaaS Datacenter* also requests through *Juju Manager* to upgrade this new State in the *Zone Synchronization Service*.

The elapsed time for steps 26-34 is defined as the time between the moment when the operating system has been completely deployed and when it has been fully upgraded, henceforth referred to as **Upgrade Time**.

4.7. EMS Deployment Phase. After the operating system is installed and upgraded, Juju EMS is installed to allow *Juju Manager* to take control of the resources in order to perform the control of the services deployed in the machine.

To this end, **(35)** *New Edge Machine* executes a script to download the *Juju Daemon* software via HTTP. **(36)** The script starts the installation and configuration, and starts the service Juju Daemon. **(37)** Juju Daemon notifies *Juju Manager* of the new status of the daemon. **(38)** This change in the status of Juju Daemon is detected by VIM Collector, which then requests *Orchestrator* to proceed with the installation.

The elapsed time for steps 25-38 is defined as the time between the moment when the machine has been fully upgraded and when the EMS Service is installed, henceforth referred to as **EMS Deployment Time**.

4.8. Service Installation Phase. Then, *Orchestrator* requests Juju to perform the automatic installation of the OpenStack Nova-Compute service together with its dependencies (hypervisor, drivers, etc.), which enables the infrastructure to provide multitenant support over virtualized resources.

To this end, **(39)** once *Orchestrator* detects that Juju Daemon is running in *New Edge Machine*, *Orchestrator* requests for a New Service Installation of OpenStack Nova-Compute. **(40)** *Juju Manager* requests the OpenStack Nova-Compute service to *Juju Catalog*. **(41)** *Juju Manager* replies to *New Edge Machine* with the Juju Charm (service template) that contains all the information to install and configure the *OpenStack Compute* service. **(42)** *New Edge Machine* starts the installation of the OpenStack Nova-Compute with all the dependencies, configuring the configuration files and starting the service using the steps indicated in the Charm. **(43)** When the OpenStack Nova-Compute service starts in the *New Edge Machine* service, it starts sending packets to inform *OpenStack Manager* that it is alive. Then, **(44)** *Juju Manager* notifies the VIM Collector and the VIM Collector pushes, through *Orchestrator*, the new state that OpenStack Nova-Compute is ready to use *Machine A* in the geographical Zone *Edge 1*. **(45)** At same time of step 44 *Juju Manager* notifies the *Zone Synchronization Service* that the OpenStack Nova-Compute is ready to use *Machine A* in the availability Zone *Edge 1*.

The elapsed time for steps 39-45 is defined as the time between the moment when the SIM takes control and when the service is installed in the physical machine, henceforth referred to as **Service Deployment Time**.

4.9. VNF Service Deployment Phase. Once the nova-compute is deployed and running, the 5G VNF service can then be installed into a Virtual Machine that will be allocated in the new hardware resource. In our case, a 5G Centralized Unit or

CU VNF service is to be deployed. This CU VNF is an open-source implementation based on OpenAirInterface [41]. This service is currently being employed in the 5G infrastructure in the EU H2020 5G-PPP SELFNET project.

To perform the installation of this 5G VFN service, **(46)** *Orchestrator* sends to *Juju Manager* a request to instantiate a New Virtual Machine in the Availability Zone Edge1. **(47)** *OpenStack Manager* notifies *Zone Synchronization Service* that Machine A is ready. **(48)** *Zone Synchronization Service* moves Machine A to Availability Zone Edge 1 to be completely synchronized with the real geographical layout as indicated by *MaaS*. **(49)** *Juju Manager* requests to instantiate a new service choosing the VNF image, the availability zone and the tenant. OpenStack supports multitenant that are different from those tenants supported by MaaS. That means that the tenants that belong to the VIM are nested over the PIM, which implies higher security in the system due to the additional isolation layer and benefits for the operational and capital expenditure. **(50)** OpenStack Manager deploys a Virtual Machine with Ubuntu 16.04 in *New Edge Machine* allocated in Edge1 called *New Virtual Machine*. **(51)** The Virtual Machine starts to spawn between *OpenStack Manager* to *New Edge Machine* and it is switched on once this process is completed. Then the new VM **(52)** starts in *New Edge Machine* with at least two interfaces as requested by the CU. These interfaces are connected to the associated network using the network manager's information provided by collecting plug-ins into *Orchestrator*. **(53)** The Virtual Machine requests to download the EMS Service (Juju Daemon). **(54)** The Virtual Machine installs the Juju Daemon, and configures and starts the service. **(55)** Once the Juju Daemon Service is ready, Juju takes control of the VNF, and notifies the SIM Collector that the Juju Daemon was successfully installed. Then, the SIM Collector pushes the new state through *Orchestrator*. **(56)** Once Juju Daemon Service is ready, it sends to *Juju Manager* its new state. **(57)** *Orchestrator* has been notified that *Juju Daemon* was installed properly in *New Virtual Machine* and *Orchestrator* requests the installation of a CU software to *New Virtual Machine*. **(58)** *New Virtual Machine* requests *Juju Catalog* to install the CU Charm in the VM. **(59)** Once the CU software is downloaded, the installation and configuration of the CU starts. **(60)** If some additional packages should be installed due to service dependencies, this will be performed as part of the installation process. After that, **(61)** the CU service starts properly, **(62)** loading all the configurations that need to be completely operative. At this state, **(63)** the 5G Service notifies Juju Manager that the installation has been properly carried out. This is done in fact by the Charm (service template). **(64)** Finally, the 5G service notifies *Orchestrator* that the installation has been completed properly.

The elapsed time for steps 46-64 is defined as the time between the moment when the service is installed in the physical machine and when the 5G CU VNF is running inside of OpenStack, henceforth referred to as **VNF Deployment Time**.

At this stage, a new 5G VNF (CU) is fully operational and running in a VNF allocated to a given tenant network in the new physical machine at the edge of the network that was at the beginning simply a bare-metal computer. The whole process is a zero-touch orchestration where there is no human intervention during the course.

4.10. Parallel Deployment Considerations. The software takes care of multiple service requests by dealing with a separate deployment status per each of the requests being processed in different threads. If a user increases the number of individual services that are working together, the deployment time increases. However, when this happens the Orchestrator controls the interdependency and the internal scheduler decides the best manner to deploy a complex service by taking in consideration the state of each of the services currently being deployed.

5. Empirical Validation and Results

5.1. Execution Testbed. The purpose of this testbed is to empirically investigate the service deployment time achieved to perform the installation of VNF services for 5G MEC infrastructures from bare-metal. The testbed has been created using 6 physical machines as managed computers, and each one has 8 cores, 24 Gbytes of RAM, and 4x1Gbps Ethernet NICs + IPMI Ethernet. Each physical machine hosts up to 8 virtual machines. Therefore, the managed infrastructure consists of up to 48 machines. These machines are managed by a physical machine with an Intel Xeon Processor E5-2630 v4 with 32GBytes and 3x10Gbit Ethernet NIC acting as a management plane depicted in the right part of Figure 1.

Architecturally, we evaluate both the centralized (conventional cloud computing) and the distributed (MEC) scenarios. In the centralized scenario, all the 48 machines belong to the datacenter core, whilst in the distributed scenario we emulate 6 edge geographical locations in our lab, each one deployed in each of the physical machines. In the distributed scenario, each edge will have its own Edge Controller as shown in the left part of Figure 1.

Each of the edge machines has been virtualized in order to be able to emulate up to 8 "physical machines" (virtual machines acting as physical machines in the infrastructure) in each of the servers used in the testbed, i.e., 48 physical machines. Thus, the testbed is making use of nested virtualization inside of each of the virtual machines that are acting as compute nodes registered in OpenStack to host virtual machines. It is known that nested virtualization has a significant impact in performance for the VNFs deployed therein. However, the main purpose of the testbed is not to optimize the performance of the virtualized service deployed but to demonstrate the scalability of the proposed architecture, so this deployment has allowed us to test the proposed architecture with a large number of managed resources. If we can show that the performance is acceptable even in this suboptimal setting of using nested virtual machines, we can expect that better performance will be achieved in a more optimal setting. It is worth mentioning that the architecture presented in Figure 1 matches the deployment carried out in our testbed.

5.2. Experimental Setup. Only the most complex scenario has been executed due to the significant time required to gather and process the results. The scenario is composed by 48 machines to be provisioned that are at the beginning of the experiment at bare-metal state (no operating system installed and even no hard disk partitioning). The experiment has been executed 12 times. Then, the experiment will initiate the creation of a new 5G CU VNF service in each of these 48 machines with a request time interval of 1 second. It means, that at time *t=1*, the first 5G CU VNF Service Deployment request will be started and in *t=2* a new one will be requested on top of the first one and so on up to 48. At the end of the experiment, 48 physical machines will be handled by OpenStack and each of them will host a CU VNF belonging to a tenant. For each of the service deployment requests initiated, the complete list of steps presented in Section 4 is executed in an orchestrated way by the proposed Orchestrator. For the sake of simplicity, this scenario has used the same tenant for all the VNFs but the prototype supports such capability. A separate experiment conducted in our lab has allowed us to realize that the use of multitenancy does not impose any overhead in the Orchestrator and in the service deployment times.

Centralized versus Distributed Infrastructures. As mentioned, this experiment has been executed over two architecturally different infrastructures that are geographically distributed across different locations to see empirically the performance comparison of a traditional centralized multitenant cloud infrastructure against the novel distributed multitenant MEC infrastructure. Between each of the execution of the experiments, a complete clean-up of the infrastructures has been carried out to allow the execution of the experiment always from bare metal as starting point.

Horizontal versus Vertical Deployment Strategies. The usage of emulated physical machines where many of them are hosted inside the same physical device allows us to also investigate how the selection order of such physical machines affects the deployment times. To analyze this factor, two different deployment strategies have been defined. The horizontal strategy will select an emulated physical machine by doing a round-robin between all the real physical device whereas the vertical strategy will select all the emulated physical machines hosted by the same real physical device before to start with the next device.

Overall, the above arrangements led to a comparative analysis of two different infrastructures to deploy 5G services (cloud computing and mobile edge computing) against the two different deployment strategies described. The results of such analysis are described in the next subsection.

5.3. Performance Results. Figure 3 shows the detailed times taken by each of the 48 CU 5G VNF services deployed in the analyzed scenarios. The top half of the figure ((a) and (b)) shows the times in the centralized cloud computing infrastructure and the bottom shows the times in the distributed MEC infrastructure. The average times taken by all the services to be deployed in each of the scenarios employed are shown in Table 2. As expected, the total service deployment times are lower for the distributed MEC architecture when compared with the centralized ones. In the centralized scenario, the datacenter controller receives all the requests and must serve all the images; whereas in the distributed scenario, the requests are served by the edge controllers, and hence this scenario has six nodes serving requests and this fact has a positive impact on the service deployment time. Figure 3 shows a white-box analysis of all the times defined in Section 4. It also shows five new times where there was a transition (indicated by the '->' symbol) between different steps along the orchestration process and the system was waiting for an event to trigger the next step.

Table 2: Average deployment times in minutes.

Scenario	Time (min)
Centralized cloud computing - horizontal deployment	47.0 ± 9.2
Centralized cloud computing - vertical deployment	55.9 ± 6.3
Distributed MEC - horizontal deployment	36.0 ± 2.7
Distributed MEC - vertical deployment	50.4 ± 2.8

In both infrastructure scenarios, the vertical deployment strategy always provided worse results than the horizontal one. In the vertical deployment strategy, each physical node starts 8 managed nodes in 8 seconds before starting other managed nodes in another physical node. This imposes a stress in the workload handled by the hypervisor and also imposes a stress burst in the network traffic generated by the simultaneous provisioning from the same edge. In contrast, in the alternative horizontal deployment strategy, a managed machine is powered on at each physical node using round-robin so that when a managed machine boots in a physical node there is a lag of six seconds until other managed node starts in the same physical node and it allows homogenizing better the performance across all the physical machines.

It is worth noting that the distributed MEC infrastructure orchestrated using a distributed horizontal strategy performs the deployment of 48 CU VNF services from bare-metal in 36 minutes on average. This is a 30% improvement in service deployment time compared with the current centralized cloud computing infrastructure. Moreover, it is noted that the horizontal deployment strategy provides significantly better results in both centralized and distributed infrastructures, 17% and 10% time savings, respectively, compared with the vertical deployment strategy.

Furthermore, Figure 3 also shows how the stability of the times gathered by all the services deployed is significantly higher for the distributed infrastructure when compared with the centralized infrastructure. This is empirically shown in the standard deviations shown in Table 2, where one can see a deviation of almost 20% and 11% for the centralized infrastructure in contrast to a much more stable 7.5% and 5.5% for the distributed infrastructures. It makes the latter architecture not only faster but also more stable against the scalability stress tests.

This change of behaviour in the standard deviation is mainly due to the efficiency of the architecture that provides a clear offloading in the datacenter to pass the responsibility of serving the operating system through the machines to the edges. This methodology not only allows offloading the main

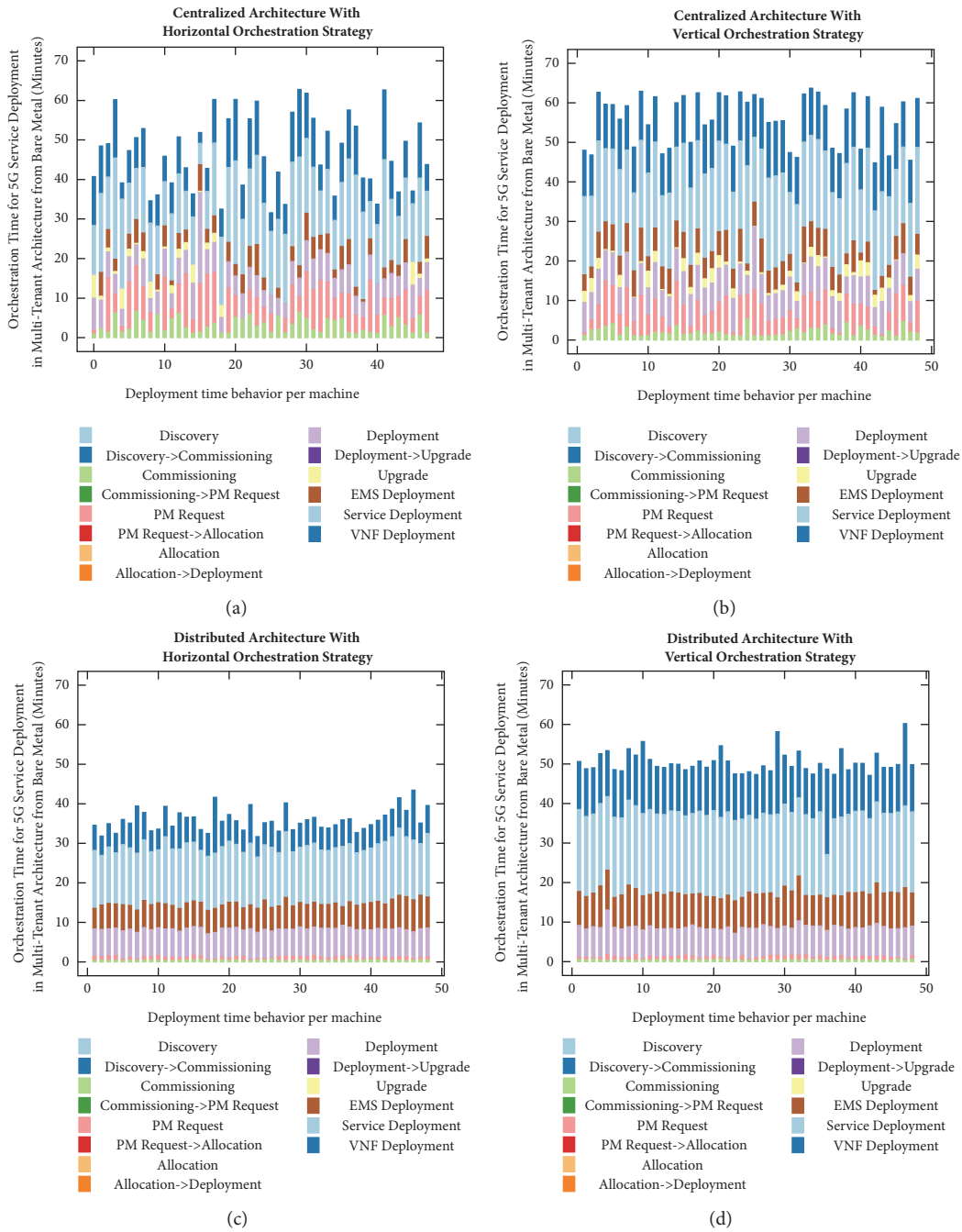

FIGURE 3: Service deployment times to deploy a 5G CU VNF service from bare-metal. (a) and (b) correspond to the centralized infrastructure. (c) and (d) correspond to the distributed infrastructure. (a, c) show the times for a horizontal deployment strategy whereas (b, d) show the vertical one.

data paths but also is able to optimize the time consumed transferring all the files required by the machines thanks to the speed of the network in each edge.

All the four experiments carried out in this testbed together have empirically validated the efficiency and effectiveness of the proposed orchestration solution, which is able to fulfill the ambitious KPI of the 5G-PPP program of deploying 5G services in less than 90 minutes. It is noted that the best case scenario (using the recommended distributed MEC infrastructure with the horizontal deployment strategy) managed to complete the deployment of 48 VNFs in 48 machines in 36 minutes and the worst scenario (using the alternative centralized cloud computing infrastructure with the vertical deployment strategy) was achieved in less than 56 minutes.

In order to gain a better understanding of how the time was spent along all the steps involved, Figure 4 shows the distribution of times of each of the phases involved in the

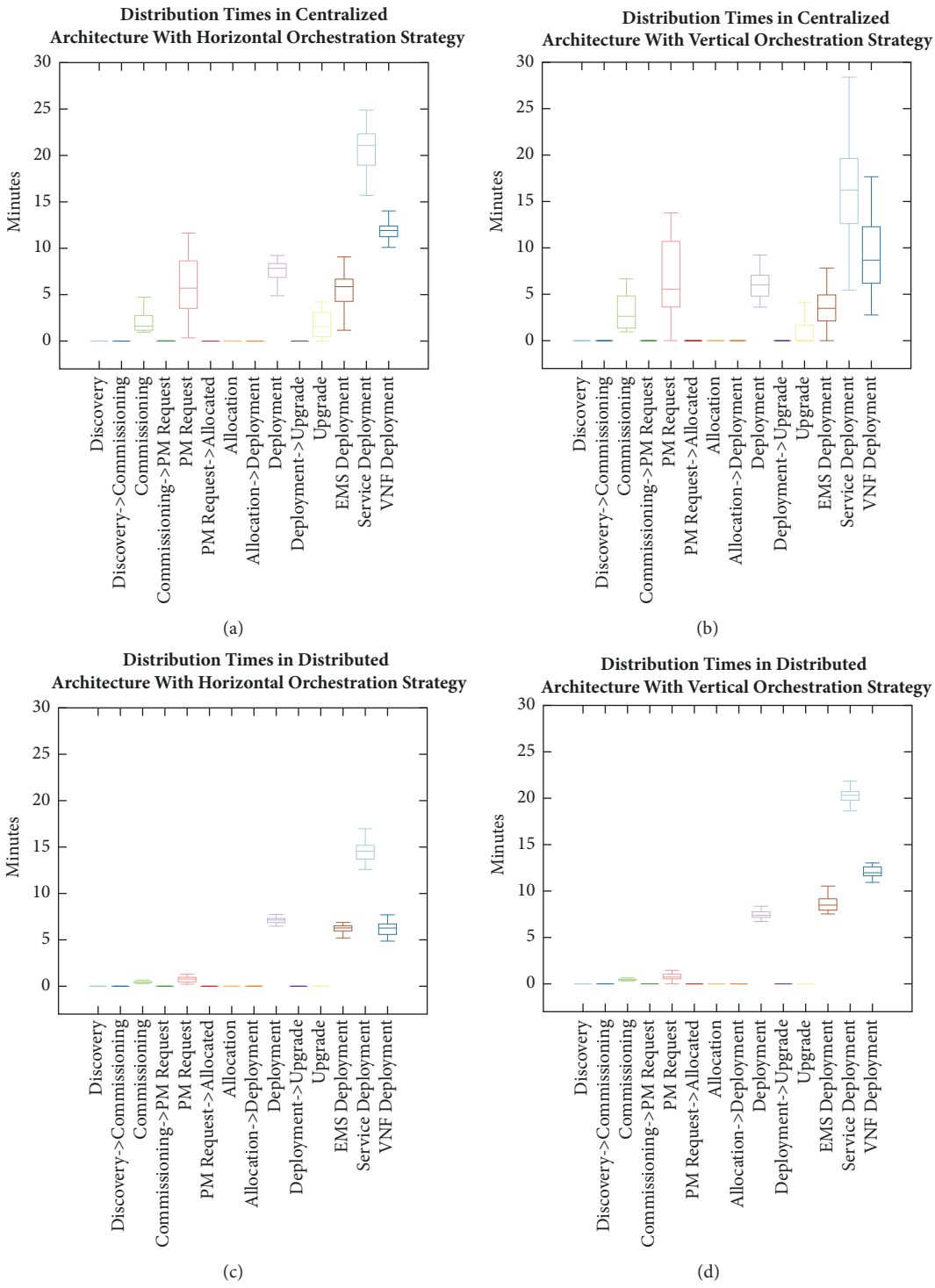

Figure 4: Distribution of the times along the orchestration steps related to the deployment of a 5G VNF CU service from bare-metal. (a) and (b) correspond to the centralized infrastructure whilst (c) and (d) correspond to the distributed infrastructure. (a, c) show times for a horizontal orchestration strategy whereas (b, d) correspond to the vertical one.

orchestration process. It reveals more insights on identifying the bottleneck points that should be further investigated in order to further reduce the overall times in future work. The subfigures shown in Figure 4 directly matches the subfigures shown in Figure 3.

Figure 4 shows that the service deployment phase is the most time-consuming portion around (around 18 mins) of all the times analyzed, followed by the VNF Service Deployment phase (around 10 mins). These are the times required to install all the OpenStack software and its dependencies into

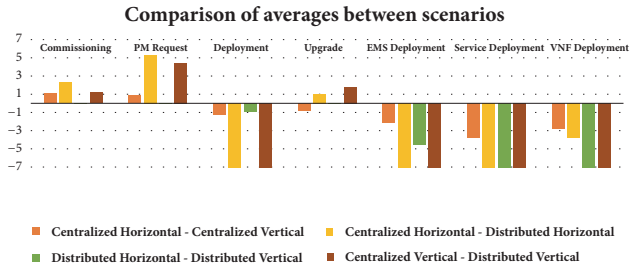

Figure 5: Differences in average service deployment times.

the physical machine and also to have the VNF deployed, installed and running. It is noted that the testbed does not have any cached VNF images in the physical machines since it is the first allocated VM with that image. It means that the spawning time, defined as the process to move the VNF image from the management image repository to the proper location, needs to be considered (only happening on the first installation of a VNF service in the physical machine). Subsequent installations of such services will take much less time since the image is already spawned.

When the figures are compared between infrastructures, it can be noticed how the commissioning time and PM request time are significantly reduced in the distributed infrastructure when compared with the centralized one. It shows the improvement related to the exploitation of the locality of the data and the distributed split of workload across different areas of the distributed architecture proposed. This improvement highlights the benefit from employing the distributed architecture.

In order to gain more insights and further identify room for potential improvement, the differences in the times of different phases among the four service deployment scenarios (2 deployment orchestration strategies per infrastructure x 2 infrastructures) are highlighted in Figure 5. It is noted that phases yielding insignificant time differences have been removed from the figure for brevity. The following four comparisons are conducted, and a difference is equal to the time of a phase in the former scenario (e.g., Centralized Horizontal) minus that of the latter scenario (Centralized Vertical). A positive difference indicates increased time in the former scenario whilst a negative one indicates decreased time.

Centralized Horizontal-Centralized Vertical. This comparison is to show differences between the two orchestrations strategies using the centralized architecture. The differences in the first phases are negligible whilst in the last three phases (highlighted in the last three light brown bars) there is 9.53 minutes of joint time reduction, which is the main reason why the Centralized Horizontal scenario is about 9 minutes quicker than the Centralized Vertical scenario as shown in Table 2.

Centralized Horizontal-Distributed Horizontal. This comparison is to show differences between architectures using the same horizontal orchestration strategy. It is noted that the only phase that reduced time in the Centralized Horizontal is the EMS Deployment (highlighted in the only negative yellow bar), which can be further investigated to benefit the recommended Distributed Horizontal approach. In all the other phases, Distributed Horizontal outperforms its alternative.

Distributed Horizontal-Distributed Vertical. This comparison is to show differences between the two orchestration strategies using the distributed architecture. It can be seen that Distributed Horizontal manages to reduce times in the three last phases (highlighted in the last three green bars), which explains the better performance of the horizontal orchestration strategy compared with its vertical counterpart in the distributed MEC infrastructure.

Centralized Vertical Distributed Vertical. This comparison is to show differences between architectures using the same vertical orchestration strategy. The only phase showing time reduction in Centralized Vertical is the EMS Deployment (highlighted in the only negative dark brown bar). Again, this phase can be further examined in Distributed Vertical for potential improvement.

6. Conclusion

This paper presents the design, prototyping, and empirical evaluation of a new 5G orchestration solution that is capable of achieving fast service deployment across different locations in a 5G Multiaccess/Mobile Edge Computing compliant infrastructure. This novel Orchestrator enables the control of each managed element in each of the three infrastructure layers (physical, virtual, and service infrastructures) through an integrated and automated orchestration process in the proposed architecture.

Consequently, this solution empowers 5G network administrators to own a complete life-cycle control of the resources at different layers in an integrated operational environment and be able to deploy a 5G service from bare-metal in less than 90 minutes in a real infrastructure (even in the disadvantaged case of employing nested virtualization), meeting one of the most ambitious KPIs proposed by the 5G-PPP. The extensive empirical results gathered from differently combined scenarios of architectures (centralized versus distributed) and orchestration strategies (horizontal versus vertical) have successfully validated the proposed solution. When the recommended distributed MEC infrastructure with the horizontal deployment strategy was employed, the system completed the deployment of 48 5G VNF services in 48 physical machines in 36 minutes.

Future work will further explore optimization mechanisms to minimize the times taken in selected bottleneck steps/phases so that the overall service deployment time can be further reduced. It is also planned to explore the benefits of using an alternative event-based architecture rather than the current polling-based architecture to explore the impact on performance.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was funded in part by the European Commission Horizon 2020 5G-PPP Program under Grant Agreement no. H2020-ICT-2016-2/761913 (SliceNet: End-to-End Cognitive Network Slicing and Slice Management Framework in Visualized Multi-Domain, Multi-Tenant 5G Networks) and by the European Commission Horizon 2020 5G-PPP Programme under Grant Agreement no. H2020-ICT-2014-2/671672-SELFNET (Framework for Self-Organized Network Management in Virtualized and Software-Defined Networks). This work has been additionally funded by the UWS 5G Video Lab project.

References

[1] R. Marco Alaez, J. M. Alcaraz Calero, F. Belqasmi, M. El-Barachi, M. Badra, and O. Alfandi, "Towards an open source architecture for multi-operator LTE core networks," *Journal of Network and Computer Applications*, vol. 75, pp. 101–109, 2016.

[2] E. Commission, "Report from the commission to the european parliament and the council, on the implementation of directive 2014/61/eu of the european parliament and of the council of 15 may 2014 on measures to reduce the cost of deploying high-speed electronic communications networks," Report, 2018.

[3] I. G. B. Yahia, "Management Plane System Definition, APIs and Interfaces," *Slicenet Deliverable D2.4*, p. 95, May 2018.

[4] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computingâ a key technology towards 5g," *ETSI white paper*, vol. 11, no. 11, pp. 1–16, 2015.

[5] "D2.6 Final Report on Programme Progress And Kpis," 2017, https://5g-ppp.eu/wp-content/uploads/2017/10/Euro-5G-D2.6_Final-report-on-programme-progress-and-KPIs.pdf.

[6] L. Kanies, "Puppet," in *LISA*, 2006.

[7] M. Marschall, *Chef Infrastructure Automation Cookbook*, Packt Publishing, 2013.

[8] D. Frost, "Using capistrano," *Linux Journal*, vol. 2009, no. 117, p. 8, 2009.

[9] M. D. Poat, J. Lauret, and W. Betts, "Configuration Management and Infrastructure Monitoring Using CFEngine and Icinga for Real-time Heterogeneous Data Taking Environment," *Journal of Physics: Conference Series*, vol. 664, no. 5, p. 052020, 2015.

[10] D. Zamboni, *Learning CFEngine 3: Automated system administration for sites of any size*, O'Reilly Media, Inc, 2012.

[11] M. Burgess and R. Ralston, "Distributed resource administration using cfengine," *Software: Practice and Experience*, vol. 27, no. 9, pp. 1083–1101, 1997.

[12] B. Karakostas, "Towards autonomic cloud configuration and deployment environments," in *Cloud and Autonomic Computing (ICCAC), 2014 International Conference on*, pp. 93–96, September 2014.

[13] L. Hochstein and R. Moser, *Ansible: Up and Running: Automating Configuration Management and Deployment the Easy Way*, O'Reilly Media, Inc., 2017.

[14] D. Merkel, "Docker: lightweight linux containers for consistent development and deployment," *Linux Journal*, vol. 2014, no. 239, p. 2, 2014.

[15] B. Hosmer, "Getting started with salt stack–the other configuration management system built with python," *Linux journal*, vol. 2012, no. 223, 2012.

[16] B. A. C. W. Stacks and A. A. C. W. Instances, "In this section," *AWS CloudFormation*, p. 104, 2015.

[17] K. Shirinkin, *Getting Started with Terraform*, Packt Publishing, 2017.

[18] H. Work, "Scalr: The auto-scaling open-source amazon ec2 effort," *TechCrunch posted Apr*, vol. 3, pp. 1–6, 2008.

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

[20] S. T. Graham and X. Liu, "Critical evaluation on jClouds and cloudify abstract APIs against EC2, Azure and HP-Cloud," in *Computer Software and Applications Conference Workshops (COMPSACW), 2014 IEEE 38th International*, pp. 510–515, July 2014.

[21] "The ZOOM project zero-touch orchestration, operations and management," 2018, https://www.tmforum.org/collaboration/zoom-project/.

[22] Price, Christofer, Rivera et al., "OPNFV: An open platform to accelerate NFV," *White Paper. A Linux Foundation Collaborative Project*, 2012.

[23] G. A. Carella and T. Magedanz, "Open baton: a framework for virtual network function management and orchestration for emerging software-based 5g networks," *Newsletter*, vol. 2016, 2015.

[24] T. Binz, U. Breitenbücher, F. Haupt et al., "OpenTOSCA – A Runtime for TOSCA-Based Cloud Applications," in *Service Oriented Computing and Applications*, vol. 8274 of *Lecture Notes in Computer Science*, pp. 692–695, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[25] J. Wettinger, T. Binz, U. Breitenbücher, O. Kopp, F. Leymann, and M. Zimmermann, "Unified invocation of scripts and services for provisioning, deployment, and management of cloud applications based on TOSCA," in *Proceedings of the 4th International Conference on Cloud Computing and Services Science, CLOSER 2014*, pp. 559–568, Spain, April 2014.

[26] M. Caballer, S. Zala, Á. L. García, G. Moltó, P. O. Fernández, and M. Velten, "Orchestrating Complex Application Architectures in Heterogeneous Clouds," *Journal of Grid Computing*, vol. 16, no. 1, pp. 3–18, 2018.

[27] M. Zimmermann, M. Falkenthal, F. Leymann, F. W. Baumann, and U. Odefey, "Automating the provisioning and integration of analytics tools with data resources in industrial environments using OpenTOSCA," in *Proceedings of the 21st IEEE International Enterprise Distributed Object Computing Conference Workshops, EDOCW 2017*, pp. 3–7, Canada, October 2017.

[28] "Etsis new zero touch network and service management group starts work," 2018, http://www.etsi.org/news-events/news/1254-2018-01-news-etsi-s-new-zero-touch-network-and-service-management-group-starts-work.

[29] K. Vukojevic-Haupt, S. G. Sáez, F. Haupt, D. Karastoyanova, and F. Leymann, "A middleware-centric optimization approach for the automated provisioning of services in the cloud," in *Proceedings of the 7th IEEE International Conference on Cloud Computing Technology and Science, CloudCom 2015*, pp. 174–179, Canada, December 2015.

[30] Y. Demchenko, S. Filiposka, R. Tuminauskas et al., "Enabling Automated Network Services Provisioning for Cloud Based

Applications Using Zero Touch Provisioning," in *IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC)*, pp. 458–464, Cyprus, December 2015.

[31] N. K. Singh, S. Thakur, H. Chaurasiya, and H. Nagdev, "Automated provisioning of application in IAAS cloud using Ansible configuration management," in *Proceedings of the 1st International Conference on Next Generation Computing Technologies, NGCT 2015*, pp. 81–85, India, September 2015.

[32] Y. Demchenko, F. Turkmen, C. De Laat, C. Blanchet, and C. Loomis, "Cloud based big data infrastructure: Architectural components and automated provisioning," in *Proceedings of the 14th International Conference on High Performance Computing and Simulation, HPCS 2016*, pp. 628–636, Austria, July 2016.

[33] D. McPherson and B. Dykes, "VLAN Aggregation for Efficient IP Address Allocation," 2001.

[34] A. Sirbu, C. Pop, and F. Pop, "MaaS advanced provisioning and reservation system," in *Proceedings of the 1st International Workshop on Automated Incident Management in Cloud*, pp. 13–18, ACM, New York, NY, USA, 2015.

[35] T. Brisco, "RFC 1794 - DNS support for load balancing - ietf tools," 1995.

[36] K. Sollins, "The tftp protocol (revision 2)," 1992.

[37] T. Slaight, "Using IPMI platform management in modular computer systems," *Intel Corporation, Intel Developer Forum*, 2003.

[38] T. Rosado and J. Bernardino, "An overview of openstack architecture," in *Proceedings of the 18th International Database Engineering & #38; Applications Symposium*, pp. 366-367, ACM, New York, NY, USA, 2014.

[39] J. Kim, D. Kim, and S. Choi, "3GPP SA2 architecture and functions for 5G mobile communication system," *ICT Express*, vol. 3, no. 1, pp. 1–8, 2017.

[40] M. Johnston and S. Venaas, "Dynamic Host Configuration Protocol (DHCP) Options for the Intel Preboot eXecution Environment (PXE)," RFC Editor RFC4578, 2006.

[41] N. Nikaein, M. K. Marina, S. Manickam, A. Dawson, R. Knopp, and C. Bonnet, "OpenAirInterface: A Flexible Platform for 5G Research," *Computer Communication Review — acm sigcomm*, vol. 44, no. 5, pp. 33–38, 2014.

[42] D. Armstrong, D. Espling, J. Tordsson, K. Djemame, and E. Elmroth, "Contextualization: Dynamic configuration of virtual machines," *Journal of Cloud Computing*, vol. 4, no. 1, 2015.

[43] D. K. Meghana and J. G. Reddy, "Cloud-based approach to increase the performance of execution of binary by using the separate debug file," in *Applied and Theoretical Computing and Communication Technology (iCATccT), 2016 2nd International Conference on IEEE*, pp. 743–746, July 2016.

[44] R. Sheu, S. Yuan, X. Liu, and P. Chung, "A plug-and-work tool for cloud system reconfiguration with single command," in *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, pp. 477–479, April 2016.

[45] M. De Bayser, L. G. Azevedo, and R. Cerqueira, "ResearchOps: The case for DevOps in scientific applications," in *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on IEEE*, pp. 1398–1404, May 2015.

[46] G. Iuhasz and I. Dragan, "An Overview of Monitoring Tools for Big Data and Cloud Applications," in *Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2015 17th International Symposium on IEEE*, pp. 363–366, September 2015.

[47] Y. Katsuno and H. Takahashi, "An automated parallel approach for rapid deployment of composite application servers," in *Cloud Engineering (IC2E), 2015 IEEE International Conference on IEEE*, pp. 126–134, March 2015.

[48] N. Forsgren and J. Humble, "DevOps: Profiles in ITSM Performance and Contributing Factors," *SSRN Electronic Journal*, 2015.

[49] J. R. Santiago, *Observability and the decision-making process in information technology service management: A delphi study [Ph.D. thesis]*, Northcentral University, 2017.

[50] K. Torberntsson and Y. Rydin, *A study of configuration management systems: Solutions for deployment and configuration of software in a cloud environment*, 2014.

[51] O. Hanappi, W. Hummer, and S. Dustdar, "Asserting reliable convergence for configuration management scripts," *ACM SIGPLAN Notices*, vol. 51, no. 10, pp. 328–343, 2016.

[52] J. Wettinger, "Gathering solutions and providing apis for their orchestration to implement continuous software delivery," 2017.

[53] S. Thakur, S. C. Gupta, N. Singh, and S. Geddam, "Mitigating and Patching System Vulnerabilities Using Ansible: A Comparative Study of Various Configuration Management Tools for IAAS Cloud," in *Information Systems Design and Intelligent Applications*, vol. 433 of *Advances in Intelligent Systems and Computing*, pp. 21–29, Springer India, New Delhi, 2016.

[54] V. Sobeslav and A. Komarek, "OpenSource Automation in Cloud Computing," in *Proceedings of the 4th International Conference on Computer Engineering and Networks*, pp. 805–812, Springer, 2015.

[55] P. Safarik and S. Schuenemann, "Ground segment as a service," in *Proceedings of the 14th International Conference on Space Operations, 2016*, Republic of Korea, May 2016.

[56] M. Falkenthal, J. Barzen, U. Breitenbücher, C. Fehling, and F. Leymann, "From pattern languages to solution implementations," in *Proceedings of the in Proceedings of the Sixth International Conferences on Pervasive Patterns and Applications (PATTERNS)*, pp. 12–21, 2014.

[57] "Efficient pattern application: validating the concept of solution implementations in different domains," *International Journal on Advances in Software*, vol. 7, 2018.

[58] H. Ding, *Persistence and discovery of reusable cloud application topologies [Master, thesis]*, 2016.

[59] M. De Lucia, J. Wray, and S. S. Collmann, "Cloud migration experiment configuration and results," Technical Report, US Army Research Laboratory Aberdeen Proving Ground United States, 2017.

[60] B. Jones, G. McCance, C. Cordeiro, D. Giordano, S. Traylen, and D. Moreno García, "Future Approach to tier-0 extension," *Journal of Physics: Conference Series*, vol. 898, p. 082040, 2017.

[61] C. Adam, N. Anerousis, M. F. Bulut et al., "Design and Evaluation of a Self-Service Delivery Framework," in *International Conference on Service-Oriented Computing*, pp. 445–452, 2017.

[62] R. Underwood, "Building bridges: The system administration tools and techniques used to deploy bridges," in *Proceedings of the Practice and Experience in Advanced Research Computing 2017 on Sustainability, Success and Impact*, p. 5, ACM, 2017.

[63] P. M. Smith, J. St. John, and S. L. Harrell, "There and Back Again," in *Proceedings of the the HPC Systems Professionals Workshop*, pp. 1–7, Denver, CO, USA, November 2017.

[64] D. Armstrong, K. Djemame, and R. Kavanagh, "Towards energy aware cloud computing application construction," *Journal of Cloud Computing*, vol. 6, no. 1, 2017.

[65] M. Airaj, "Enable cloud DevOps approach for industry and higher education," *Concurrency Computation*, vol. 29, no. 5, 2017.

[66] M. K. Aljundi, *Tools and practices to enhance DevOps core values*, 2018, Master Thesis For LUT university, 2018.

[67] A. Aguinaga Mendibil, "Orquestación de servicios vagrant y capistrano," *Grado en Ingeniera en Tecnologa de Telecomunicacin, Telekomunikazio Teknologiaren Ingeniaritzako Gradua*, 2017.

[68] S. Son, H. Choi, B. T. Oh, S. W. Kim, and B. S. Kim, "Cloud SLA relationships in multi-cloud environment: models and practices," in *Proceedings of the 8th International Conference on Computer Modeling*, pp. 1–6, ACM, Canberra, Australia.

[69] P. Raj and A. Raman, "Multi-cloud management: Technologies, tools, and techniques," in *Software-Defined Cloud Centers*, 240, p. 219, Springer, 2018.

[70] L. Wang and X. V. Wang, "Latest advancement in cloud technologies," in *Cloud-Based Cyber-Physical Systems in Manufacturing*, pp. 3–31, Springer, 2018.

[71] M. Villari, A. Celesti, G. Tricomi, A. Galletta, and M. Fazio, "Deployment orchestration of microservices with geographical constraints for Edge computing," in *Proceedings of the 2017 IEEE Symposium on Computers and Communications, ISCC 2017*, pp. 633–638, Greece, July 2017.

[72] P. Massonet, A. Levin, A. Celesti, and M. Villari, "BEACON project: Software defined security service function chaining in federated clouds," Advances in Service-Oriented and Cloud Computing: Workshops of ESOCC 2016, Vienna,Austria, September 5–7, 2016, Revised Selected Papers, vol. 707, Springer, p. 305, 2018.

[73] M. Villari, G. Tricomi, A. Celesti, and M. Fazio, "Orchestration for the Deployment of Distributed Applications with Geographical Constraints in Cloud Federation," in *Cloud Infrastructures, Services, and IoT Systems for Smart Cities*, pp. 177–187, Springer, 2017.

[74] P. Raj and A. Raman, "Automated Multi-cloud Operations and Container Orchestration," in *Software-Defined Cloud Centers*, pp. 185–218, Springer, 2018.

[75] S. Chatlapalle, "Generic Deployment Tools for Telecom Apps in Cloud," 2018.

[76] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On Multi-Access Edge Computing: A Survey of the Emerging 5G Network Edge Cloud Architecture and Orchestration," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1657–1681, 2017.

[77] A. Komarek, J. Pavlik, L. Mercl, and V. Sobeslav, "VNF Orchestration and Modeling with ETSI MANO Compliant Frameworks," in *Internet of Things, Smart Spaces, and Next Generation Networks and Systems*, vol. 10531 of *Lecture Notes in Computer Science*, pp. 121–131, Springer International Publishing, Cham, 2017.

[78] J. Santos, T. Wauters, B. Volckaert, and F. de Turck, "Fog computing: Enabling the management and orchestration of smart city applications in 5G networks," *Entropy*, vol. 20, no. 1, 2018.