This project has received funding from the European's Union Horizon 2020 research innovation programme under Grant Agreement No. 957258

Architecture for Scalable, Self-*, human-centric, Intelligent, Secure, and Tactile next generation IoT

ASSIST-IoT Technical Report #9

Evolution of MANO towards the Cloud-Native Paradigm for the Edge Computing

Alejandro Fornés-Leal, Ignacio Lacalle, Rafael Vaño, Carlos E. Palau, Fernando Boronat, Maria Ganzha and Marcin Paprzycki

2022 International conference on Advanced Computing and Intelligent Technologies (ICACIT 2022)

Evolution of MANO towards the Cloud-Native Paradigm for the Edge Computing

Alejandro Fornés-Leal[1][0000-0001-5914-6045], Ignacio Lacalle[1][0000-0002-6002-4050], Rafael Vaño[1][0000-0003-2372-6253], Carlos E. Palau[1][0000-0002-3795-5404], Fernando Boronat[1][0000-0001-5525-3441], Maria Ganzha[2][0000-0001-7714-4844] and Marcin Paprzycki[2][0000-0002-8069-2152]

[1] Communications Department, Universitat Politècnica de València, Valencia, Spain
[2] Systems Research Institute Polish Academy of Sciences, Warszawa, Poland
alforlea@upv.es, iglaub@upv.es

Abstract. The Management and Orchestration framework (MANO) is the main element of the Network Function Virtualization paradigm. It is in charge of managing the lifecycle of virtualized functions, from instantiation to manageability, live configuration and termination. This kind of framework was originally designed to orchestrate network functions over virtual machines. However, the Cloud-Native approach, based on containers and microservices, has emerged and needs to be included as a part of MANO, to leverage all the inherent benefits that it brings. This contribution identifies the key enablers that have to be addressed, from the MANO perspective, to fully exploit the capabilities and to obtain real added value from implementing this novel approach, focusing mainly on resource-constrained environments. Besides, an analysis of current status of open-source frameworks aiming at the Cloud-Native adaptation is presented, showing that while Cloud-Native approaches vís-a-vis network functions are widely accepted (at least, by the research community), there is still room for further research and integration.

Keywords: MANO, Cloud-Native, NFV, Edge Computing.

Introduction

Edge computing and Network Function Virtualization (NFV) have been two of the main technological paradigms of the IT environments that emerged during past years. The former is focused on bringing computation capabilities as close to the source of data (and actuation) as possible, hence optimizing network bandwidth usage, supporting low-latency applications and reducing privacy and security breaches in contrast to traditional cloud computing models [1]. The latter (NFV) aims at virtualizing network functions, facilitating their instantiation on general purpose equipment, thus decoupling the provided services from the hardware that deliver them [2].

The European Telecommunications Standards Institute (ETSI) defined an architectural framework for NFV Management and Orchestration (MANO) [3], with the objective of facilitating the management of the lifecycle of the Virtualized Network Functions (VNFs), from their instantiation to their configuration and termination. This framework has had a great acceptance not only by researchers but also within the business scope, due to great possibilities that it brings for developing new applications and business models.

However, MANO initially relied on Virtual Machines (VMs) for delivering the intended functionalities. Therefore, it was focused mostly on cloud infrastructures, which do not integrate well with the current move towards the Cloud-Native approach. Rather than referring to the place where applications are instantiated, Cloud-Native is about the way in which they are created and deployed [4], aiming at increasing their speed (of development and deployment), offering better scaling, and leveraging only the required hardware resources.

Current research has shown that offering the VNFs with containers rather that with VMs can provide great benefits, while opening the execution environment possibilities. Specifically, it can enable managing and orchestrating network (and non-network) virtualized functions in hardware with less resources. This, in turn, can promote emergence of new (or improved) use cases and business models. However, Cloud-Native solutions have to be integrated within MANO frameworks to be useful.

This contribution focuses on the current status of the integration of NFV and the Cloud-Native approach for edge distributed deployments, analyzing the key enablers and potential future of this technological symbiosis. The remainder of this paper is organized as follows: Section 2 presents a review of MANO, edge computing and the Cloud-Native model; Section 3 focuses on the key enablers and technologies for unlocking the adoption of Cloud-Native Virtualized Network Functions (CNFs); afterwards, in Section 4 an analysis of the current MANO solutions is presented, whereas in Section 5 an analysis of barriers and potential future of this paradigm transformation is considered. Finally, conclusions are drawn in Section 6.

MANO framework and Edge within IoT

ETSI MANO for NFV

ETSI was selected to host the Industry Specification Group for Network Function Virtualization (ETSI ISG NFV). Apart from the MANO architectural framework, the initial documents include an overview of the infrastructure, descriptions of the network, hypervisor, computing domains, and additional aspects such as security and trust, resilience, and quality of service. Based on these documents, the MANO architecture is presented in Fig. 1.

NFV requires access to hardware computing, storage, and network resources, which are provided by the NFV Infrastructure (NFVI) and assigned to the VNFs, depending on the specific demands. VNFs can be managed locally by the Element Management System (EMS).

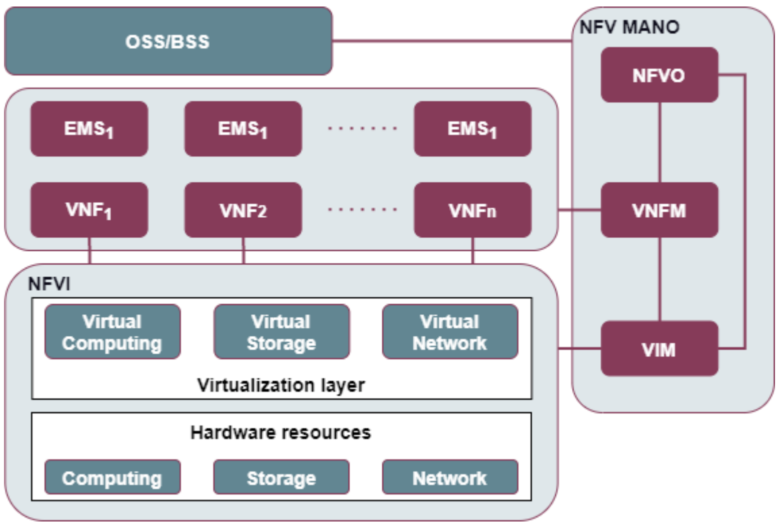

Fig. 1. MANO architecture.

All the hardware/virtual systems and the virtualized functions are managed by the NFV Management and Orchestration (MANO), that is composed of three elements: the NFV Orchestrator (NFVO), the VNF Manager (VNFM), and the Virtualised Infrastructure Manager (VIM).

The NFVI is managed by the VIM. Here, computation, storage and network-related resources are assigned to the virtual resources needed by specific functions.

A VNF accesses its respective resources, globally configured, and supervised by the VNFM component. The VNFM also performs the respective coordination and adaptation role for configuration and event reporting between the VIM and the EMS.

The NFVO is responsible of connecting or combining NFVs as building blocks, managing orchestration of NFVI resources across multiple VIMs and lifecycle management of Network Services (NSs).

Edge computing and NFV

Since the beginning, edge computing has been divided into three main categories of implementation: Multi-access Edge Computing (MEC), cloudlets, and fog computing [5]. All of them share the vision of edge computing and strongly rely on mechanisms such as virtualization, safety resources management and metering. However, there are clear differences in configuration, characteristics and scope. In summary:

Multi-access (formerly, mobile) edge computing is associated with Radio Access Network (RAN), where the edge capabilities are located at the base stations. This implementation is oriented towards ISP providers and is focused on achieving edge computing benefits for 4/5G use cases.Cloudlets can be understood as “replicas” of the cloud capabilities but closer to the edge of the network, thus reducing latency, round-trip time and backhaul bandwidth consumption. They are conceived as "cloud in a box", acting as cloud running over one, or a cluster of, resource-rich server(s).

Fog computing (FC), instead, aims at leveraging the flexibility of IoT, to perform edge computing functions. Using "fog nodes", that can be spanned through the edge-to-cloud continuum creating 1 to N "near-end" layers or tiers, FC orchestrates their functioning to take advantage of wide range of devices in the continuum of the spectrum.

The combination of NFV and edge implementations provides great benefits, especially when considering distributed environments such as Industrial IoT. First of all, NFV allows instantiating and modifying existing services much faster, leveraging general-purpose equipment and avoiding the need of deploying dedicated hardware. Moreover, NFV enables the possibility of deploying and configuring services automatically, which can be further enhanced via intent-based methods [6], or self-organizing techniques [7]. Besides, NFV enables network slicing, so both network bandwidth and latency can be optimized, boosting the inherent latency reduction provided by the edge computing paradigm. Hence, it is clearly visible how NFV came into play with the objective of reducing both the time and cost related to initial deployments and operations, i.e., for reducing CAPEX and OPEX.

Benefits of the Cloud-Native approach for NFV

As aforementioned, edge deployments can benefit from NFV, when compared to the use of dedicated hardware equipment. Still, as will be argued in what follows, the NFV model can be further improved if extended with the Cloud-Native approach, based on microservices and containers rather than virtual machines and traditional software architectures:

Reducing (further) development and operational costs: Cloud Native applications are based on a set of granular, small microservices, which can be developed, deployed and optimized independently, improving software DevOps cycles (avoiding having to package a complete, monolithic solution). At the edge, where hardware resources can be limited, containers reduce the amount of overhead required by the VMs, resulting in an effective reduction of costs.

Improving the agility of a system: In contrast to VMs, containers (that host the microservices) are much faster to deploy, substitute and scale. The latter is a key feature in an edge environment since containers are flexible, using only the required resources and leaving space for other applications.

Novel business paradigm: The paradigm shifts towards renting additional servers on-demand, rather than acquiring a fixed set of VMs in advance. This model entails reduction of costs to both infrastructure operators and to end users. In addition, less storage is needed, as Cloud-Native paradigm pushes towards stateless microservices.

Cloud-Native MANO enablers

Evolving from VNFs to CNFs is not just as trivial as changing the VIM from OpenStack (or a similar infrastructure management technology) to a container orchestrator platform, such as Kubernetes (k8s). This is particularly the case if this evolution is intended to extract all the potential brought by the Cloud-Native approach. The key enablers that MANO and CNFs need to support are presented in this section.

Microservices

The Cloud-Native approach follows an architecture design paradigm based on microservices. It provides a robust solution as a set of small, loosely-coupled, independent services, which are isolated in small coherent and autonomous units, to solve the problem of complex architectures and code redundancies. Microservices architecture allows scaling, or updating, each service without affecting the rest of the services of the system.

Although Service-Oriented Architecture could be adopted, this paradigm is not as flexible as microservices, especially in those cases where the development team is spread out, the components have clear functionality boundaries, and if components can be potentially reused for other applications: microservices extract the full potential of DevOps cycles

Containerization

It is a lightweight, agile virtualization alternative to VMs. A container packages all the software needed to run a single application or microservice, including all code, libraries, and required dependencies. They are smaller, faster and more portable than VMs, since they do not require including guest Operative System (OS) in each instance, leveraging the host kernel (OS virtualization), instead of the virtualized hardware infrastructure (as the VMs do). Here, despite the fact that *Docker* is currently the most popular container engine, there are also alternatives, such as *CRI-O*, *Containerd* or *runc*.

Apart from containers, another lightweight virtualization technology that is worth to mention, is *unikernels*. Unikernels are similar to virtual machines, but without many of the inherent services from the OS, leaving just the ones that are actually needed for executing their application [8]. In principle, they are more difficult to design than VMs or containers, but they have at least the same potential of performance as containers plus improved security and isolation features, which can be a valuable aspect in distributed environments (lesser attack surfaces).

Container Orchestration Technologies

VIMs, like OpenStack, OpenVIM, or other available vendor alternatives, have been designed for managing virtualized hardware resources to deploy VMs, not for managing virtualized OS spaces, therefore they are not valid for orchestrating containers: specific container orchestration technologies are needed, among which one can find k8s, the *de facto* standard, *Docker Swarm*, or *Apache Mesos*.

Apart from *k8s*, lightweight alternatives based on it such as *k3s* or *MicroK8s* are also very interesting for distributed, resource-constrained environments, like Industrial IoT.

Packaging and management of functions: Helm charts and Juju charts

Both *Helm* and *Juju* are package and operations managers for k8s. A Helm chart is a collection of files that describes a set of k8s resources, and that can be used for deploying either an application or a component of a larger application. It provides *templating*, which allows users to declare variables and use functions to modify parameters of the applications (in this case, of CNFs).

Despite *Helm* charts being more widespread, *Juju* charts, based on hooks (typically written as shell scripts) are claimed to be a more scalable tool and more effective for complex container lifecycle operations.

Networking

Typical networking schemas that apply for VMs, are not valid for containers. This is due to the fact that containers are processes that share the kernel of the host system. Therefore, from the outside world perspective, they have a common IP address.

Currently, there are two solutions for addressing this issue: Container Network Model (CNM) and Container Network Interface (CNI). Here, the latter is close to becoming the *de facto* standard, despite the fact the former is being supported by Docker.

A CNI is a plugin that implements a network interface within the container namespace, assigning an IP address to it and setting the required bridges with the host. There are different technologies for supporting networking, like *Flannel*, *Canal* and *Weave*. Networking technologies can address different connectivity aspects, from L2/L3 networking, VXLAN, Overlay and BGP, which can be of great interest for improving the NSs deployed and enable more efficient schemas of multi-cluster networking.

Service mesh and Service Discovery

Service mesh is a software infrastructure layer for controlling the communication between services. In a similar way to SDN, it decouples the control plane from the data plane. The latter is implemented as proxies on top of microservices, transparent to the business functionality, whereas the former interacts with proxies to provide different functionalities related to connectivity (service discovery, load balancing, dynamic routing control), security (encryption, policy enforcement) and observability (alerting based on traffic alerts).

In contrast to the networking section, service mesh provides application level features. Here, among the most popular tools, one can find *Istio*, *Linkerd* and *Consul*.

Besides, service discovery mechanisms are required when large number of microservices are available. To that end, a dedicated database of services (i.e., service registry) should be in place and expose the location of services (IPs addresses, ports) when queried by rightful users/services. *Consul* and *CoreDNS* are examples of service discovery engines.

Communication bus

Microservices require communication and exchange of information for delivering a standalone service or application. In general, REST APIs are enough to handle it. Still, some network services may require larger information and telemetry data exchange. Here, a dedicated bus (e.g., *Apache Kafka*) may be needed for providing scalable, reliable and resilient communication.

Bare-metal deployment

Another deployment paradigm that is recently evolving, consists in running container orchestration platforms in bare metal, instead of a virtualized infrastructure. It involves executing Cloud-Native applications in containers running directly on the hardware, which results in a great simplification of network setup. Getting rid of the virtualized infrastructure layer comes with many benefits among which one can find: increase of the available resources, since less overhead for virtualization is needed (can be key for resource-constrained environments); improvement of the operational performance; and reduction of costs related to licenses.

Nevertheless, MANO systems are designed to communicated to NFVI, so integration effort is required in most cases.

Hybrid Network Services Support

The emergence of CNFs does not involve elimination of VNFs or Physical Network Functions (PNFs, provided by specialized hardware) from the MANO ecosystem; at least not yet.

Great effort has been put into designing and implementing VNFs. However, migrating them to CNFs is not a trivial task, especially for big appliances (for instance, a firewall with all the functionalities it provides). Hence, MANO should have the capability of managing all network function types, either Physical, Virtual (on top of a VIM) and Cloud-Native ones (leveraging a container orchestrator platform). Nevertheless, over time, full migration to Cloud-Native should occur, as “edge world” can’t afford to have two computing environments.

Current status of MANO solutions

In this section, a summary of the current status of different open-source MANO frameworks, for supporting CNFs, is provided. Although in different maturity levels, all of them support CNF creation and management, illustrating the already-achieved pervasiveness of Cloud-Native approaches (at least, as considered from the research sector), despite the fact that further effort is needed, especially as what concerns standardization and integration activities.

Tacker

Tacker is a generic VNFM and an NFVO from OpenStack, based on ETSI MANO, developed to operate VNFs and compose NSs on an infrastructure platform like OpenStack or Kubernetes. Tacker proposes an architecture, in which Kubernetes acts as a VIM, parallel to OpenStack (see Fig. 2). Tacker is composed by a set of drivers to act over both VIMs: (1) *infra*, responsible of the operations to operate the VIMs; (2) *vim*, in charge of their registration; (3) *mgmt*, which facilitates the configuration of VNFs; (4) *monitor*, responsible for executing actions towards that end; and (5) *policy*, for VNF operations based on policies. Support for k8s has only been provided for the first two plugins.

However, the information regarding different Cloud-Native aspects is quite low, from both official documentation and existing literature. No reference to packaging and management functions, networking or service mesh is provided, so any of these functionalities have to be provided outside of the scope of MANO.

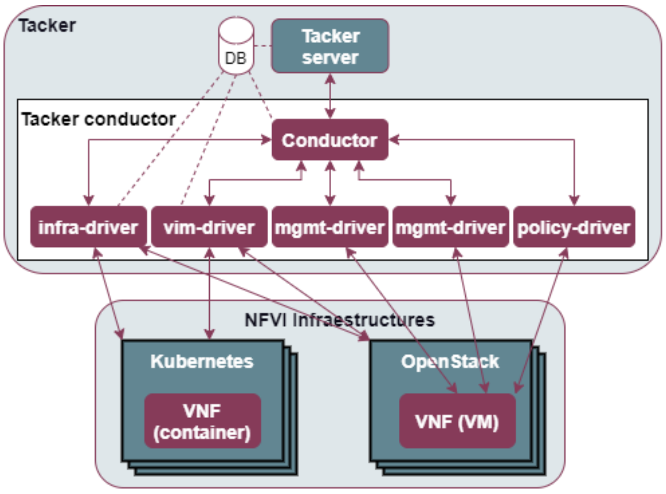

Fig. 2. Tacker architecture.

OSM

OSM is the MANO framework developed under the ETSI umbrella. To manage CNFs, OSM is able to register *k8s* clusters as long as they are connected to an OpenStack-like VIM, as depicted in Fig. 3, allowing hybrid deployments (they can be connected to a “dummy” VIM, in which case those hybrid deployments are not possible).

OSM provides support for both *Helm* and *Juju*, which allow not only deploying but also configuring CNFs via primitives using charts or charms. Some features related to *k8s* networking, and service mesh, are not implemented yet, as a part of the framework (these features can be managed externally). This is hindering the orchestration for managing different clusters in multi-domain environments. This problem can be addressed by connecting each of them to VIMs to manage inter-cluster networking.

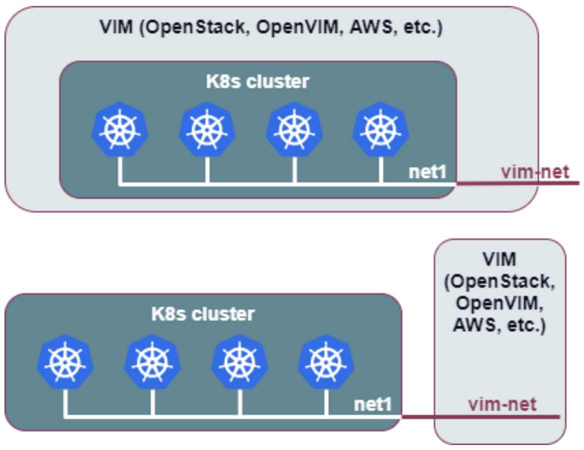
VIM (OpenStack, OpenVIM, AWS, etc.)
K8s cluster
net1
vim-net
VIM (OpenStack, OpenVIM, AWS, etc.)
K8s cluster
net1
vim-net

Fig. 3. Interaction between VIMs and Kubernetes in OSM.

Anuket

Anuket is based on the merge between OPNFV and the CNTT, and developed under the umbrella of the Linux Foundation. It is one of the actions that push towards fully exploiting all Cloud-Native capabilities. It defines a Reference Architecture (RA, see Fig. 4) and a set of requirements to be fulfilled during its development and implementation. Up to this moment, it supports only the first of the two defined RAs, with basic features regarding CNF instantiation (e.g., Helm chart packaging is not yet supported).

The project is currently working on fully integrating Cloud-Native approach, from the networking perspective to *k8s* add-ons management (service mesh, monitoring, logging, tracing, etc.). It shows great potential and is worthy further monitoring.

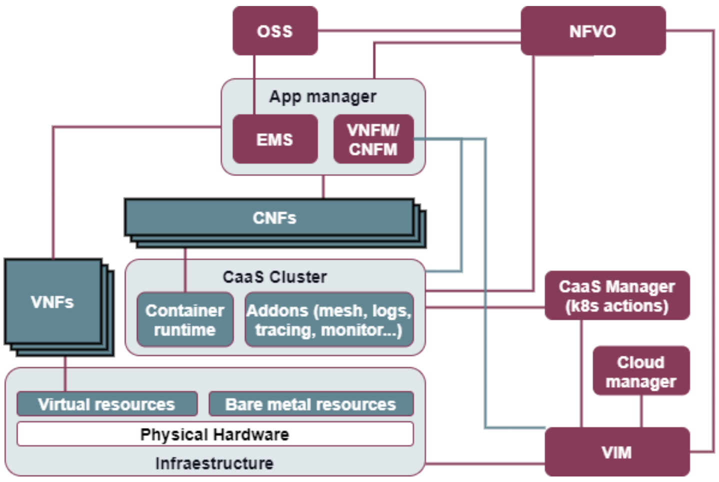

Fig. 4. Anuket architecture.

ONAP

ONAP is one of the largest automation frameworks, composed of different subsystems, covering much more aspects than the rest to create an end-to-end platform. ETSI MANO architecture blocks can be mapped to ONAP ones, although alignment work is still being done to make it fully compliant.

ONAP allows packaging, deploying and configuring CNFs with Helm charts. Current version contains dedicated APIs for creating and modifying k8s resource templates, check services' health status and communicating with telemetry tools, while also supporting hybrid and multi-cluster environments. Its future role regarding CNF networking, inventory and service mesh enablers still needs to be evaluated.

Other MANO frameworks

Let us now briefly summarize two other MANO frameworks. SONATA was one of the first MANO frameworks providing support to CNFs on top of k8s. It includes CNF lifecycle support and descriptor validation, allows defining and collecting custom metrics from both CNFs and k8s infrastructure, and implementing hybrid network schemas. Unlike OSM, SONATA has a plugin to connect directly to k8s VIM (as well as ONAP and Tacker's vim driver).

Besides, Cloudify is an open-source solution that goes beyond MANO purposes, similarly to ONAP. It has evolved differently to other alternatives to integrate multiple cloud environments (many platforms supported e.g., k8s, Azure, AWS, etc.), and Cloud-Native features into the network orchestration model, aligned but not fully embracing MANO specifications. Among its features, Cloudify includes instantiation and configuration of CNFs, intent placement based on policies, networking among clusters and hybrid NSs support, being one of the most advanced open-source solution

EMCO

The Edge Multi-Cluster Orchestrator (EMCO), previously known as ONAP4K8S, is a different framework compared to the previous ones, as it is not following the MANO specifications (see Fig. 5). It is designed for deploying and orchestrating only Cloud-Native applications over a set of *k8s* clusters, from cloud to edge (hence multi-cloud but not supporting hybrid network schemas). This framework has been leveraged by two vendor open-source projects, Intel *Openness* and Aarna Networks *AMCOP*.

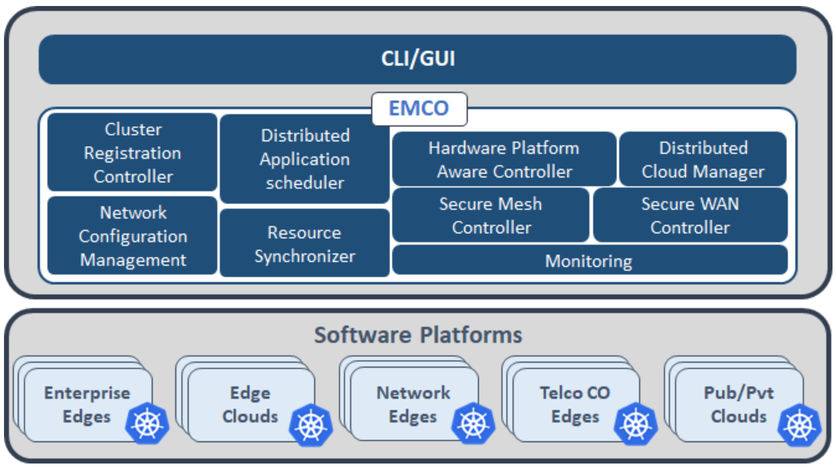

Fig. 5. EMCO architecture.

Being focused purely on Cloud-Native approach, the functionalities provided for CNFs are much more advanced in comparison to other frameworks (e.g., mesh network supported with Istio, better analytics gathering and service discovery features), while being more adapted to the edge environments. They allow hybrid deployments (VNF and PNF support), although without following MANO specifications, which in the long term may cause some interoperability issues.

Summarizing what has been described thus far, in Table 1, main features of noted frameworks have been presented.

Table 1. Comparative table between MANO frameworks.

Feature	**MANO frameworks**							
	Tacker	OSM	Anuket	ONAP	SONATA	Cloudify	EMCO	AMCOP
MANO compliance	Yes	Yes	Yes	Partially	Yes	Partially	No	No
CNF onboarding	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
CNF validation	Proposed	Yes	Yes	Yes	Yes	Yes	No	Yes
CNF Helm/Juju support	No	Yes	No	Yes	No	Yes	Yes	Yes
CNF monitoring	No	No	Proposed	Yes	Yes	Yes	Yes	Yes

CNI-based networking	No	No	Proposed	No	No	Yes	Yes	Yes
Service mesh	No	No	Proposed	No	No	No	Yes	Yes
Hybrid deployment support	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes
Multi-cluster k8s support	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Discussion and expected outcomes

This section aims at outlining the set of existing barriers, as well as the expected evolution of MANO frameworks, towards achieving full Cloud-Native support. Here, note that the state of evolution among the different frameworks depends on different aspects. First of all, those solutions that have been present for some years need more adaptation effort to optimally accommodate CNFs. This becomes more challenging if they strictly follow the MANO specifications, since specifications towards the embracement of Cloud-Native additional features are not yet finalized (including the role of some of the enablers identified in Section 3).

On the contrary, novel solutions such as EMCO (and other non-open-source solutions) have greater potential as they are very agile embracing novel Cloud-Native solutions. Still, specifications are needed so the NFV ecosystem keeps the high interoperability level that was brought by ETSI MANO.

Among the current initiatives (and besides ETSI MANO), Anuket is the one promoting harder a standard RA that includes both VNFs and CNFs, deployed either over k8s and OpenStack-like VIMs.

To the authors' expectations, hybrid deployments will be the norm during current decade, since there has been a great effort put into developing VNFs and deploying complete virtualized infrastructures, which will not be immediately substituted by the Cloud-Native paradigm (besides, some VNFs are very challenging to be containerized into CNFs). Further explanation of the expected evolution has been presented in [4], and is summarized in Fig. 6.

In the future, a model purely based on kubernetes, for both VNFs and CNFs, leveraging technologies like *KubeVirt* or *Virtlet* to deploy those network functions that couldn't be deployed as containers is foreseen.

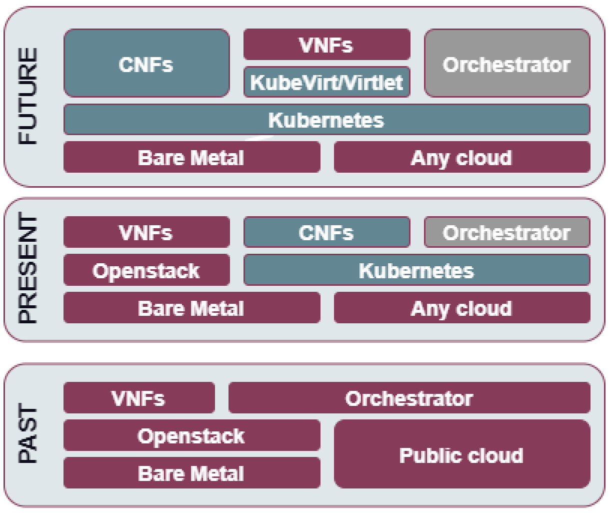

Fig. 6. Expected evolution of MANO towards the Cloud-Native approach.

Cloud-Native approach promises great benefits, like an automated installation and configuration of CNFs; dynamic scaling according to workload; self-healing and fault-tolerant reliable mechanisms; automated performance monitoring; high reusability and portability, etc. [9]. However, as any novel paradigm, it faces challenges and barriers to overcome, such as trust issues over administrative domains; isolation and security of CNFs (containers are less secure than VMs); network function chaining; adaptability of existing VNFs; change from the software development point of view, moving towards microservices approach; business adaptation delay needed for transforming current model towards Cloud-Native; and necessity needs for further research and, most importantly, of standardization actions [4], [9].

In some implementations, although CNF onboarding, configurations and metrics retrieval can be performed via MANO, Cloud-Native capabilities are not managed by the framework. Features such as container networking, service mesh (layer 7 or even novel layer 3, IP models) and service discovery are left outside the scope of it, and are expected to be managed externally via auxiliary tools. This diminishes the potential that can be obtained from integrating those features within the scope of MANO, especially in the networking area, in the same manner than SDN and WAN schemas were introduced, in the past, for managing NFV networks.

Concluding remarks

This paper provides a comprehensive evaluation of the key enablers for bringing the Cloud-Native benefits to the NFV ecosystem, especially for edge deployments, along with the current evolution of existing MANO frameworks and expected barriers and future developments related to this approach.

There are different aspects related to container orchestration, such as service mesh, networking, service discovery, etc. that have not yet been addressed directly by the existing frameworks, especially by those that follow the MANO specifications.

This shows the necessity of advancing current NFV standardization activities towards Cloud-Native, effort that, for instance, the Anuket project is pushing forward with its Reference Architecture. In this context, there are already solutions that fully embrace the Cloud-Native model, integrating many of its features while keeping support for traditional VNFs and for composing hybrid network services. On the one hand, they show the benefits that can be extracted from it, but on the other hand, the lack of standards is very likely to cause interoperability problems in the long term.

Hybrid services, combining VNFs, CNFs and PNFs, are expected to remain for many years. Cloud-Native brings many benefits, but it is still a novel, and hence an immature concept.

Further work is needed not just for integrating existing MANO frameworks with Cloud-Native technologies, but for delivering specifications agreed by all the actors involved in the NFV ecosystem, as well as for attracting both developers and market towards this new paradigm.

Acknowledgment. This work is part of ASSIST-IoT project, that has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement 957258.

References

K. Cao, Y. Liu, G. Meng, and Q. Sun: An Overview on Edge Computing Research. IEEE Access (8),85714–85728 (2020).

C. Tipantuna and P. Yanchapaxi: Network functions virtualization: An overview and open-source projects. In: 2017 IEEE 2nd Ecuador Tech. Chapters Meet. ETCM 2017, vol. 2017-January, pp. 1–6 (2018).

ETSI: GS NFV-MAN 001 Network Functions Virtualisation (NFV); Management and Orchestration (2014).

5G-PPP Software Network Working Group: Cloud-Native and Verticals' services (2019).

K. Dolui and S. K. Datta: Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing. In: GIoTS 2017 - Glob. Internet Things Summit, Proc., LNCS (2017).

E. Zeydan and Y. Turk: Recent Advances in Intent-Based Networking: A Survey. IEEE Veh. Technol. Conf. (2020-May) (2020).

P. V. Klaine, M. A. Imran, O. Onireti, and R. D. Souza: A Survey of Machine Learning Techniques Applied to Self-Organizing Cellular Networks. IEEE Commun. Surv. Tutorials (19). 2392–2431 (2017).

5G-PPP Technology Board Working Group and 5G-IA's Trials Working Group: Edge Computing for 5G Networks (2021).

S. D. A. Shah, M. A. Gregory, and S. Li: Cloud-Native Network Slicing Using Software Defined Networking Based Multi-Access Edge Computing: A Survey. IEEE Access (9). 10903–10924 (2021).