Multi-access Edge Computing as a Service

Pedro Escaleira, Miguel Mota, Diogo Gomes, João P. Barraca, Rui L. Aguiar
Instituto de Telecomunicações
Universidade de Aveiro
Aveiro, Portugal
{escaleira,miguel.mota,dgomes,jpbarraca,ruilaa}@av.it.pt

***Abstract*—Standardization organizations, such as the European Telecommunications Standards Institute (ETSI), have been gathering efforts to specify the Edge Computing paradigm. However, there is still a lack of complete, interface-wise, actual implementations and evaluations of a fully functional Edge Computing architecture. On these grounds, the work presented in this paper proposes a new Multi-access Edge Computing (MEC)-Network Functions Virtualization (NFV) architecture for a challenging Business to Business to Consumer (B2B2C) model, based on the references provided by ETSI, and provides a prototype implementation to demonstrate its viability. The tests conducted show that the proposed framework can be efficiently deployed, allowing Telecommunications Operators to rapidly instantiate and provide an elastic Edge Infrastructure to their customers.**

***Index Terms*—Multi-access Edge Computing (MEC), Network Functions Virtualization (NFV), Edge as a Service (EaaS), Open Source MANO (OSM), Virtual Network Function (VNF), Business to Business to Consumer (B2B2C)**

I. Introduction

Over the last few years, the world has witnessed the proliferation of Cloud Computing (CC). It allowed the transition from costly, on-premises servers, not able to dynamically scale computing resources and requiring constant management, to a highly scalable, available, adaptable, and self-awareness computation model. The strategy was to allow customers to deploy their solutions to a cluster of distributed data centers [1].

However, as latency-sensitive services started to be deployed, a considerable amount of research activities started to focus on new technologies that "extend the Cloud" to the network's Edge, closer to where the User Equiments (UEs), which produce the data to be computed, are located. One of these technologies is Multi-access Edge Computing (MEC), which has been conceptualized by the European Telecommunications Standards Institute (ETSI) since 2014 [2].

The concept of MEC has evolved over the last few years, and recently ETSI proposed the integration between MEC and Network Functions Virtualization (NFV) architectures, due to overlapping components and use cases [3]. NFV is a paradigm aiming to decouple Network Functions (NFs) from their physical equipment, and leverage existing virtualization technologies to virtualize them [4]. This integration between both concepts followed the first suggestion made by Sciancalepore *et al.* [5], which argued that this combination could be performed thanks to a superposition between the management and orchestration systems of both technologies. That has the benefit of reducing the Capital Expenditures (CAPEX) and Operating Expenses (OPEX), by allowing the reuse of the NFV framework that Telco Operators might already possess.

To this end, and taking into consideration ETSI's references for MEC and NFV [3], we propose an enhanced MEC-NFV architecture for a challenging Business to Business to Consumer (B2B2C) use case, and demonstrate its feasibility in a Proof of Concept (PoC) development based on Open Source MANO (OSM), as the NFV-Management and Orchestration (MANO), and Kubernetes, as the Edge Host container orchestrator. Furthermore, the presented work introduces the possibility of a MEC environment being provided on demand by a Network Service Provider (NSP) as a service to their costumers, the Communication Service Providers (CSPs), hence enabling a MEC as a Service (MECaaS) model.

The rest of this document is structured as follows. First, we review the underlying concepts of the proposed architecture in Section II, as well as some relevant works related to the conceptualization of a MEC-NFV architecture. In Section III, we present our MEC-NFV architecture, having as a basis the ETSI reference architecture. Then in Section IV, we describe our PoC, and later in Section V, we evaluate its performance in terms of deployment and scaling time. Finally, we conclude our research in Section VI, with a summary and the main future directions related to the proposed solution.

II. Motivation and Use Case

In this work, we propose a MECaaS architecture, where the main focus is to give a response to a Business to Business to Consumer (B2B2C) model. The considered scenario has a Telco Operator, the NSP, providing other Telco Operators, the CSPs with a MEC platform for their customers' applications. This is different from the typical use case that most Telco Operators have, where their immediate customers are the ones that want to deploy the Edge Applications and not other Telco Operators i.e., usually Telco Operators have a Business to Consumer (B2C) model, being both a NSP and a CSP. This most common model is also the one considered by ETSI while developing the MEC reference framework architecture. With this information in mind, there was the need for enhancing the original architecture provided by ETSI.

A. Virtualization in the Edge

When it comes to virtualization approaches, two main types are relevant to consider: hardware-level virtualization and the OS-level virtualization [6].

The hardware-level virtualization is characterized by the virtualization of the hardware of a machine. In practice, this means that applications can be packed with an OS, and then distributed and deployed in several machines with different underlying hardware or OSs, using Virtual Machines (VMs). By contrast, containerization, or OS-level virtualization, is a different means of distributing applications in a lightweight way where there is no need to copy an entire OS as with the VMs, since when a container is launched, the host OS will run the analogous application in a controlled environment.

These differences also manifest in how the corresponding applications can be migrated. Since a container is not packed together with an entire OS, its migration will have a smaller footprint, both in memory and in launching time, than in a VM. Nonetheless, because of the difference on how both are packaged, containerization has the disadvantage that the state is lost. There are, however, already some projects and tools, such as CRIU, which are making efforts to save and restore the state of a container, although they are still limited.

One of the requirements in MEC is that the corresponding applications are deployed as close as possible to the UEs, to decrease latency. However, if the migration time between Edge servers takes too long, UEs will notice an increase in latency and the downtime of the MEC application they are communicating with. Specific dynamics may result in not favoring VM migration if movement is limited [7]. Another issue we need to consider is that Edge servers are not meant to be as powerful as the ones on Cloud, due to real-estate and energy limitations, so the deployment of VM-based applications would consume too many valuable resources.

With these factors in mind, we have considered the deployment of containerized-based applications in the Edge, so this will be reflected in the proposed architecture.

B. ETSI NFV Containerization Considerations

In the previous subsection, we highlighted the need for containerization on the Edge. However, NFV was originally conceptualized with hardware-level virtualization in mind.

ETSI is showing interest on aligning the NFV conceptualization with Containerized VNFs (CNFs) since 2016, with the release of the document *ETSI GS NFV-EVE 004 V1.1.1* [8]. Since then, it released some Interfaces and Architectures (IFAs) regarding how the integration should be made. The most important for this work was the *ETSI GR NFV-IFA 029 V3.3.1* [9], where ETSI proposed the addition of two new functional blocks to the NFV-MANO: the Container Infrastructure Service (CIS), which is where containers are executed; and the Container Infrastructure Service Management (CISM), which is needed to manage the CIS and the containers running on it.

Although these specifications have already been released, the NFV-MANO reference architecture [10] still does not consider them and, by extension, the MEC reference does also not integrate them. Since our solution was idealized as allowing the deployment of container-based MEC Applications, and since MEC Applications are viewed as Virtual Network Functions (VNFs) in the MEC-NFV reference architecture, we considered this set of ETSI discussions while conceptualizing our MEC-NFV architecture.

C. Other MEC-NFV Proposed Solutions

Besides the first conceptualization of the integration between MEC and NFV [5], there are some other noteworthy works, which also presented a combined architecture for MEC-NFV [11]–[14], based on the one standardized by ETSI.

The authors Baldoni *et al.* [11] presented a conjunction orchestration solution between NFV and MEC, based on the 5GCity project's orchestration system. To achieve that, the authors developed the MEC Application Orchestrator (MEAO) and the MEC Platform Manager (MEPM) as new extensions on top of Eclipse fog05, a Virtualised Infrastructure Manager (VIM) specifically designed for Edge Computing use cases. Then, the MEAO is described as being integrated with the NFV Orchestrator (NFVO) in a singular top-level multi-orchestrator, which orchestrates all the VNFs, including the MEC Applications and the MEC Platform (MEP), deployed as VNFs. Although this article tries to give answers to some open issues related to the integration of MEC and NFV, it lacks the practical demonstration of the proposed solution.

In [12], Cattaneo *et al.* also proposed an architectural solution. Some relevant issues targeted in this paper were related to how the interactions between the MANO entities and the MEC ones can be conducted, taking into account the existing NFV connection points. More specifically, the authors argue that the interfaces *Mv1*, *Mv2* and *Mv3*, are almost interchangeable with the NFV's *Os-Ma-nfvo*, *Ve-Vnfm-em* and *Ve-Vnfm-vnf* connection points, since their functionalities are similar. Then, as with the previous work, they defined the MEP and the MEC Applications as being VNFs, controlled by a NFV-MANO, which in this work, was the Open Baton. In the end, the proposed architecture was tested in an Edge environment, using a CPU-intensive MEC Application.

Open Baton MANO was also used in the research presented by Carella *et al.* [13], but in a different scenario: instead of using VM-based VNFs, the authors used CNFs. To do so, they did several modifications to Open Baton, creating a new VIM driver, to manage the Docker Engine container runtime, and modifying the VNF Manager (VNFM) accordingly. In our view, this approach poses some issues, with the main one being that an existing and largely tested container orchestrator, such as Kubernetes, could have been used. Kubernetes inclusively can manage and orchestrate container runtimes other than Docker, as a CISM, and the provided management Application Programming Interface (API) can be used as a connection point between the CISM and the MANO. Adopting this limited solution results in the need to create a CISM for containers from scratch, which can be challenging and is out of the scope of a NFV-MANO framework. That also means that adding support for other container runtimes can be a challenge.

At last, Fondo-Ferreiro *et al.* [14] proposed a similar approach to the previous one but based on OSM. In it, the authors implemented a new VIM's adapter to integrate OpenNESS as the MEC framework, instead of Docker, on top of fog05. The

OpenNESS framework is a project hosted by Intel that, in practice, can be viewed as a Kubernetes flavor with specific enhancements to make it Edge-ready. An open issue described by the authors was that their solution still lacks interoperability between the MANO and MEC frameworks. By this fact, the scope of this work is valid but limited.

The main highlights of our work, in distinction with previous ones in the literature, are mainly:

- The MEAO, MEPM and MEP can be deployed as VNFs;
- The container orchestrator/manager and infrastructure i.e., the CISM and CIS, respectively, can also be simply deployed as VNF, being managed by a top-level MANO;
- The NFV-MANO used in MEC can be divided into two different MANOs, where one of them orchestrates and manages the other one, which in turn is used to enable the management and orchestration of the MEC Applications;
- The integration of the ETSI's proposed, but not yet integrated and tested on its references, CISM and CIS components into the NFV's and MEC's architecture.

III. Architecture Proposal

The central focus of this research document is to describe our MEC solution for a B2B2C scenario. That means that we needed not only to come up with an architecture that would respond to some of the other requirements we had, such as the usage of CNFs, but also to adopt the ETSI's MEC-NFV proposed architecture framework [3] for this specific use case. The proposed architecture can be analyzed in Figure 1.

Since in the B2B2C MEC scenario there are two business levels, one belonging to the NSP, and the other to the CSP, we needed to find a way of translating this fact to the architecture itself. In reality, what this means, is that the platform which the NSP provides to the CSP needed to be separated from the platform that the CSP will provide to its clients. This required the management and orchestration of the platform that controls the MEC Applications to be different from the platform that controls that one. Therefore, we have divided the MANO from the original ETSI's architectural framework into two distinct ones, as depicted in Figure 1.

The *Main MANO* is the central entity, being controlled by the NSP, and allows the multiple existing CSPs to request the set up of a new platform, which will be used by their clients to access the Edge resources. To achieve that, the said CSP can first access the Operations Support System (OSS) directly connected to the Main NFVO through an already existing NFV connection point. When this request is made i.e., when the CSP demands a new Edge platform, which then will provide to its customers, the Network Service (NS) presented in the architecture is automatically instantiated. If the MANO in question is deployed using containers, instead of VMs, the *Main MANO* will also deploy a separated VNF, containing the CISM and the CIS, as referenced in the ETSI NFV specifications presented in Section II-B. In this specific case, the new CIS is dynamically linked to the existing NFV Infrastructure (NFVI), and the CISM will be automatically connected to the MANO's NFVO and to the MEC Life Cycle Management (LCM) VNFM, to enable them to command the CISM to launch the necessary CNFs to deploy the new MANO and the MEC functional blocks.

The NS, instantiated from the referred demand made by the CSP, will contain four different VNFs: one for the new MANO, another for the MEC functional blocks, and other two for the Edge's CISM and CIS. We named this new MANO as *Edge MANO*, due to it being responsible for orchestrating and managing the MEC Applications deployed in the MEC Host (MEH). Then, the *Edge MANO* functional blocks are connected to the MEH entities through the standardized MEC-NFV connection points. This set of functional blocks is also instantiated using a separated VNF, being that the necessary connections between them and the MANO's components can be done through one or multiple internal Virtual Links (VLs) inside the referred NS. Finally, the CIS in the MEH and the CISM in the *Edge MANO* are also deployed as two separated VNFs, being the latter connected to the Edge's NFVO and VNFM and to the CIS in the MEH, to command it to deploy the necessary MEC Applications, as CNFs. The reason for that CISM and CIS being viewed as two different VNFs, instead of being grouped in the same one, is because they needed to be deployed in two distinct environments i.e., the CIS needs to be deployed in an Edge Host, while the CISM is deployed in the NSP's Cloud infrastructure, together with the other presented entities. This is also the reason why there is the need for at least two distinct VIMs: one to control the NFVI in the Cloud Infrastructure and the other the analogous Edge Host's NFVI.

In Figure 1, it is also indicated that the VNFs for the *Edge MANO* and MEC functional blocks can be CNFs i.e., depending on the targeted virtualization technique used to create this set of entities, they can be deployed as VMs or as containers. This information is important mostly because the infrastructure and corresponding manager used to deploy them will be different: for VNFs, the NFVI and VIM will be targeted, while for CNFs, the CIS and CISM will be used. And that is the main reason why there is the need to first launch the VNF with the CIS and CISM in the NSP's Cloud, in order to then launch the MEC functional blocks and the *Edge MANO* as CNFs, if that is the case.

Another important aspect of the proposed architecture is that every VIM used by the *Edge MANO* must be first added to the *Main MANO*, since this one needs to first deploy the VNF with the CIS in the corresponding MEH, and only then can the *Edge MANO* use the analogous CISM to deploy the necessary MEC Applications on top of that CIS. With regards to the MEC-NFV reference points, which are the *Mv1*, *Mv2* and *Mv3*, we agree with the argument made by Cattaneo *et al.* [12]. As stated by these authors, most of these connection points requirements are already met by the NFV *Os-Ma-nfvo*, *Ve-Vnfm-em* and *Ve-Vnfm-vnf* interfaces, respectively. So, we think that there is only the need to the ETSI's MEC and NFV Industry Specification Group (ISG) groups to agree upon the necessary additions to these existing interfaces, to enable them to meet the MEC specifications. Regarding the figure's presented connection points, there are also some

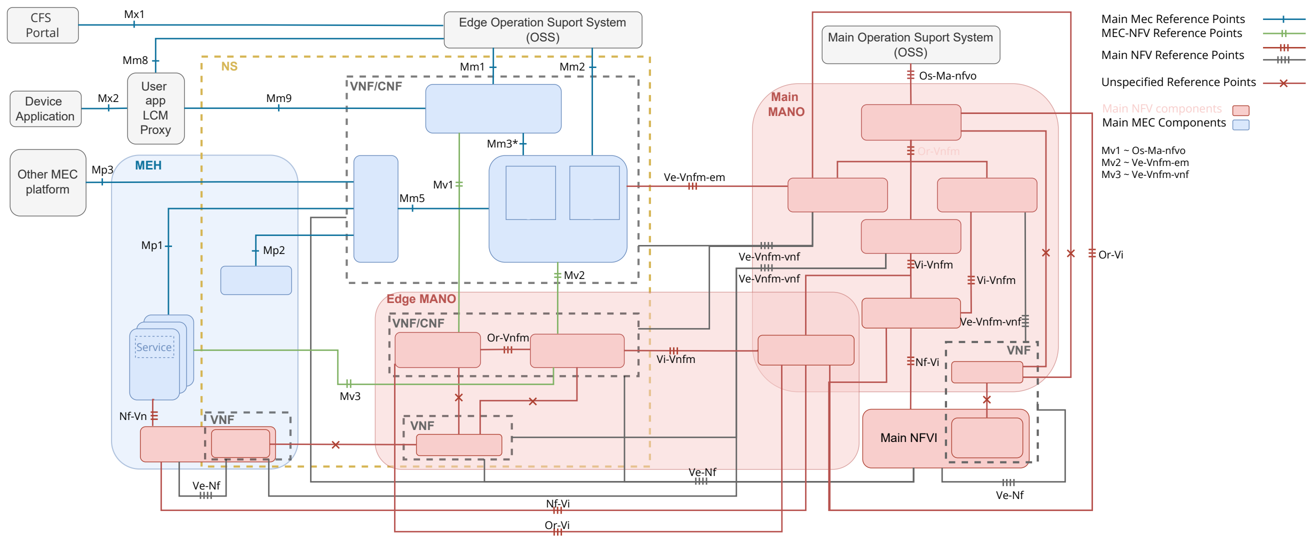

Fig. 1. B2B2C MEC-NFV proposed architecture.

labeled as *Unspecified Reference Points*. These interfaces are corresponding to the CISM and CIS connections, to which the ETSI NFV specification did just described where they would be placed, but there is still lack of their actual specification, including their nomenclature.

Finally, each CSP's customer will be able to deploy their MEC Applications through the OSS connected to the newly instantiated MANO environment. This OSS, the Customer Facing Service (CFS) Portal, and the User Application LCM Proxy could also be instantiated using a purposely designed VNF, which would then allow to connect them to the necessary entities, as depicted in Figure 1. However, the decisions behind how they could be set up are out of the scope of this document.

A. *Generalized Use Case*

Although the architecture proposed in the previous section was meant specifically for our B2B2C use case, we believe that the needed changes to fit it in a generic use case, i.e., for a Business to Consumer (B2C) model, would not be complex.

We argue that the only needed modification is the removal of the *Edge MANO*, being that the *Main MANO*, in that scenario, would conduct all the tasks previously assigned to the other one, besides all the tasks that it was already in charge of. Therefore, the NS that previously held a VNF for the MEC functional blocks and a VNF for the *Edge MANO* would now only consist of the first mentioned VNF.

All of the other components introduced for the B2B2C use case are still valid in this scenario, including the usage of two distinct VIMs and VIMs. With this in mind, all the discussions related to the connection points provided in the previous subsection also apply in this case.

IV. Proof of Concept Implementation

In order to validate our proposed architecture, we have implemented a PoC with the multiple discussed components. One important note to consider is that we tested this prototype in a private, on-premises, Cloud, instead of in a real Edge environment. For this reason, we have used only one VIM, to simplify our test scenario. Therefore, the used VIM served as both the VIM for the Main and Edge environments.

As for the software tools employed, we used version 12 of OSM, both for the *Main* and the *Edge MANOs*, OpenStack as the VIM, and Kubernetes as the CISM, since it is supported by OSM as the environment where we can deploy CNFs.

A. *Main MANO Setup*

We used OSM as the NFV-MANO to validate our proposed architecture. The OSM corresponding to the *Main MANO* was instantiated in a Kubernetes single-node cluster, which in turn was deployed in a VM.

Then, a new VIM with the credentials for our on-premises OpenStack was added to this OSM, in order to instantiate the VNFs containing the containerized environment, which will be presented further.

B. *VNF for the Containerized Environment*

We have previously referred that we used Kubernetes as our containerized environment. This decision originated from the fact that OSM already possesses a connector that communicates with the Kubernetes API, allowing it to deploy a given CNF's containers in this kind of cluster orchestrator. Also, using Kubernetes is more aligned with what was specified by ETSI in regards to the integration of this type of virtualization. We can think of the Kubernetes Control Plane as being the CISM, exposing the Kubernetes API in order for our MANO to manage the necessary resources to deploy the CNFs, and the Kubernetes Nodes as the CIS, which is where the containers are executed, being orchestrated and managed by the Kubernetes Control Plane. We also observed that Kubernetes is becoming the de-facto orchestrating solution for containers, and will have an important role in future deployments.

With this in mind, and following the proposed architecture, we created the necessary descriptors to launch a Kubernetes cluster, with any number of Nodes, as a VNF, as depicted in Figure 2. Accordingly, there are two Virtualisation Deployment Units (VDUs), each one corresponding to the two needed

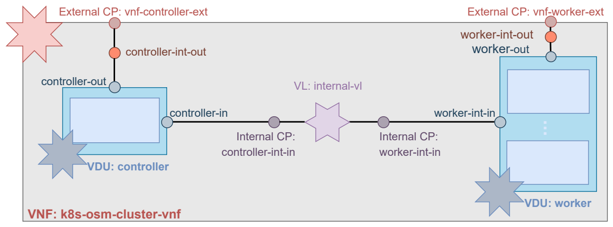

Fig. 2. VNF for the Kubernetes cluster.

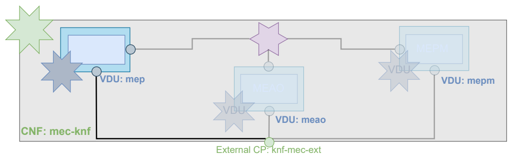

Fig. 3. CNF for the MEC functional blocks.

entities specified by ETSI for the usage of a containerized environment: the *controller*, which VDU corresponds to the CISM, and the *worker* VDU, which corresponds to the CIS. All the necessary software was installed in each VDU instance using purposely created Juju Day 1 and Day 2 scripts, which were connected to each analogous VM through Secure Shell (SSH), to execute the necessary commands.

In our PoC, this VNF has an internal VL, which allows the Workers and the Controller (corresponding to the Kubernetes Control Plane) communicate within a private sub-network. Then, both VDUs for the Controller and Worker Nodes have a network interface to “the outside”, which allows exposing the Kubernetes API, in the Controller’s case, and to expose the necessary CNF Services’, in the Worker Nodes’ case.

Apart from the option of deciding *a priori* the number of Worker Nodes, we also implemented the possibility of scaling them up and down after instantiation, allowing the customer to adapt the cluster to the computing load at each moment.

Although according to our architecture, two distinct VNFs are needed within the Edge environment, one for the CISM and the other for the CIS, we decided to only use one for this PoC since, as referred to in the beginning of the section, we have done our tests in a Cloud environment, with only one VIM, meaning that it is unnecessary, in this particular case, to have one VNF in each separated VIM.

C. CNF for the MEC Functional Blocks

The main MEC functional blocks are also instantiated and grouped in a singular VNF/CNF. In our PoC, we decided to adopt containerization to develop and deploy these entities, hence being instantiated through a CNF, as depicted in Figure 3. However, and for testing purposes, we only have implemented some of the MEP features, leaving the rest and the other two entities to be implemented in the future.

The MEP was developed to be compliant with the ETSI specification *ETSI GS MEC 011 V2.2.1* [15]. It was built with our underlying MEC system in mind and, as such, we refrained from using software that was not already used by the OSM project. Thus, the MEP was built using the CherryPy web framework and reuses the OSM’s MongoDB for data storage.

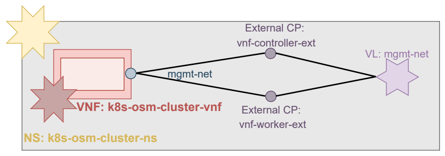

Fig. 4. NS for the Kubernetes cluster.

Due to the nature of our Kubernetes-based containerization approach and the available methods for deploying CNFs in OSM, we opted for a Helm Chart-based deployment for our MEP implementation. Therefore, our Helm Chart contains a Deployment template, to create the actual MEP pod, and a Service template, to expose the service to other MEC applications and/or UEs outside the cluster.

D. CNF for the Edge MANO

Considering that OSM is, by default, instantiated in a containerized environment, usually within a Kubernetes cluster, we had to create a CNF to deploy it. Furthermore, we found out that the OSM installation can be achieved through a Juju Bundle, called Charmed OSM, and since OSM does also support CNFs based on Juju Bundles, we decided to use the one provided, and just create the necessary CNF descriptor.

E. NS for the Containerized Environment

As defined in Section III, the proposed architecture specifies the usage of a containerized environment within the MANO i.e., a CISM and a CIS, launched by, and then connected to, the *Main MANO*, when we need to instantiate the *Edge MANO* and/or the MEC functional blocks as CNFs, instead of VNFs. Since this is the case with our PoC, as presented in the above subsections, both the OSM, as well as the MEP we have developed, are deployed using a set of containers, inside the analogous CNFs, we need to first generate the required containerized environment and link it to our *Main MANO*. This was done by including the VNF for the Kubernetes cluster we have developed, discussed in Section IV-B, in a NS purposely set up for this case, as demonstrated in Figure 4.

The *K8s Cluster* block and the two *External Connection Points (CPs)* correspond to the Kubernetes cluster VNF and its *External CPs*, previously presented in Figure 2. Then, they are connected to the NS’s VL, in order to expose the corresponding network interfaces to the “external world”.

F. NS for the Edge Environment

In Section III, we also defined that when the CSP Operator requests a new Edge environment, the *Main MANO* will instantiate a NS with the *Edge MANO*, the MEC environment, and the containerized environment for the MEH. Therefore, we have built the NS represented in Figure 5, which includes the VNFs presented in Section IV-B and the CNFs presented in the Sections IV-C and IV-D.

This NS also possesses a VL, to which each VNF/CNF attaches to, that allows all of these blocks both to connect between themselves and externally expose their services.

Both CNFs are deployed in a Kubernetes cluster, which first needs to be instantiated using the NS presented in

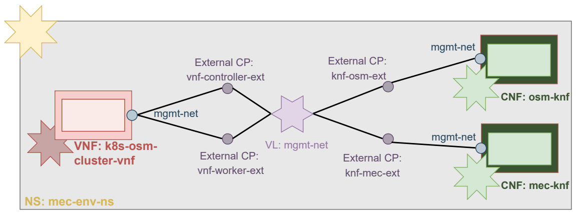

Fig. 5. NS for the Edge environment.

Section IV-E. As for the *K8s Cluster* VNF, it is installed using the *Main MANO's* only VIM. When its deployment finishes with success, it is automatically added to the *Charmed OSM*, allowing it to use this cluster as the environment to launch the necessary MEC Applications as CNFs.

Another design presupposition we considered was the possibility for a CSP to request access to the resources of a new specific MEH when it needs to. To do so, there is only the need to scale the *K8s Cluster's* VNF, and specify the *Main MANO* to instantiate the new one on a specific Edge Host. When the scaling operation finishes, the new cluster is also dynamically added to the *Charmed OSM* CNF as a new CISM.

G. *MEC Application Prototype*

In order to test our environment and consequently our MEP, we reused the University of Bologna (UNIBO)'s MEC Application [1], with some slight changes [2]. This MEC Application is a simple web application that communicates with the MEP via the *Mp1* reference point, specified by ETSI [15], to carry out actions related to its normal execution.

V. Evaluation

Intending to evaluate the discussed PoC, we performed some benchmark tests regarding the time needed to deploy all the discussed components in the previous section. This is relevant, as our focus is to enable the rapid deployment of customized MEC infrastructures, tailored for specific verticals, that can be deployed on-demand.

In order to conduct our experiments, we built a component that iteratively commanded the OSM corresponding to the *Main MANO* to instantiate both the *k8s-osm-cluster-ns* and *mec-env-ns* NSs, one after the other was successfully deployed. Then, to obtain the deployment times we will present, we obtained the analogous data OSM inserted in its Kafka storage platform. In the OSM relative to the *Main MANO*, we obtained the timestamps for the *k8s-osm-cluster-ns* and *mec-env-ns* NSs, while in the OSM related to the *Edge MANO*, we were able to gather the deployment times for our sample MEC Application, corresponding to the *mec-app-ns* NS. At the end of each iteration, the environment was cleaned up, and the next iteration is started.

In addition to the deployment time tests, we also performed tests related to the scaling time of a Worker Node of the *k8s-osm-cluster-vnf* VNF, which is the VNFs which installs the Kubernetes cluster in both *k8s-osm-cluster-ns* and *mec-env-ns* NSs. The procedure taken to conduct them was similar to the

TABLE I
Deployment time of *k8s-osm-cluster-ns*, *mec-env-ns* and *mec-app-ns* NSs in seconds.

Iteration	***k8s-osm-cluster-ns***	***mec-env-ns***	***mec-app-ns***
1	645.379	694.092	23.748
2	609.118	624.022	23.408
3	590.846	625.757	26.735
4	604.013	621.548	14.284
5	610.317	647.877	13.546
6	620.876	640.988	25.567
7	603.956	632.424	10.118
8	649.633	722.172	27.506
9	595.262	649.685	11.319
10	598.537	824.577	24.530
11	589.482	622.892	20.656
12	591.904	646.561	17.874
13	550.755	676.135	16.018
14	518.451	627.311	15.961
15	554.039	499.759	16.014
16	556.444	637.285	16.071
17	553.474	613.791	15.894
18	559.409	647.689	16.039
19	552.639	670.708	15.990
20	551.210	607.105	16.009
Average	585.287 ± 7.72	646.619 ± 13.39	18.364 ± 1.15

aforementioned for the deployment time tests, where we used a component to iteratively request the scaling out and in of the said VNF in each iteration.

To have a basis to compare the obtained results against, we did also gather the necessary times needed to deploy a dummy VNF and CNF. The considered VNF was composed by a VM with similar characteristics to the ones our VNFs instantiated, and possessed a simple Day 1 script, which when executed, simply connected to the analogous VM through SSH. As for the CNF, it was based on a Helm Chart, with a Kubernetes Service and Deployment, which when instantiated, deployed an empty Alpine Linux container. The averages obtained for these baseline tests, for 23 iterations, were approximately 4.88 minutes for the VNF, and 16.4 seconds for the CNF. All of these results and the ones presented throughout this section can be analyzed in the project's associated repository [3].

A. *VMs Specifications*

The OSM corresponding to the *Main MANO* was deployed in a Kubernetes single-node cluster, as referred in Section IV-A. In its turn, the cluster was installed on a VM with 8 VCPUs, 12 GiBs of RAM and 40 GiBs of disk space.

As also referred to in the previous section, we added an OpenStack-based VIM to this OSM, where both NSs were deployed in each iteration of our tests. In our experimentations, we only deployed one instance of each NS, and in each one of them, one instance of the *K8s Cluster* VNF. Each one of these clusters was composed of 2 nodes, each one related to a VM with 4 VCPUs and 8 GBs of RAM.

B. *Results*

1) Instantiation: With the discussed methodology and VMs specifications, we executed the referred tests script for 20 iterations. The obtained results for the deployment times of the *k8s-osm-cluster-ns*, the *mec-env-ns* and the *mec-app-ns* NSs, as well as their average value, can be consulted in Table I.

From the values presented in Table I, we can conclude that the deployment of the *k8s-osm-cluster-ns* NS takes, most of the time, roughly 10 minutes, while the *mec-env-ns* NS usually takes longer, about 11 minutes. The explanation for this small,

[1]https://github.com/berdav/unibo-test-mec
[2]https://github.com/ATNoG/mp1-test-app-mec
[3]https://github.com/ATNoG/netedge-mec

TABLE II
SCALING TIME OF *k8s-osm-cluster-vnf* VNF IN SECONDS.

Iteration	Scale-out	Scale-in
1	541.441	42.821
2	533.693	42.709
3	493.195	37.481
4	482.115	32.479
5	479.422	37.552
6	377.000	37.574
7	492.717	37.574
Average	485.655 ± 20.34	38.31 ± 1.3

but existing, difference, can be because the second NS has to deploy more VNFs/CNFs than the first one i.e., while the first only has to instantiate a Kubernetes Cluster, the second also needs to deploy the other two CNFs depicted in Figure 5. Nevertheless, these results demonstrate that a NSP can provide to a CSP Operator a complete MEC environment, from scratch, within not much more than 20 minutes.

As for our sample MEC Application, it takes, on average, about 18 seconds to be deployed. The results also show that, although our environment is inherently complex, that complexity does not negatively impact the deployment time of MEC Applications. These values are, however, representative of a simple MEC App and are expected to increase, similarly to any containerized environment, with more complex applications.

When comparing the obtained results with the baseline values for the deployment of a VNF and a CNF, we can conclude that the deployment time of both the *k8s-osm-cluster-ns* and *mec-env-ns* NSs took just over double the average sample VNF, while our MEC Application took approximately 2 more seconds when compared with the base CNF.

2) Scaling: For the scaling time tests, we run the script for 7 iterations, meaning that there were 7 scale-out and 7 scale-in operations. The obtained results can be observed in Table II.

The scale-out operation of deploying a new Kubernetes Worker took about 8 minutes on average, which in comparison with the time needed to deploy the complete cluster, done by the instantiation of the *k8s-osm-cluster-ns* NS, is 2 minutes less. In practice, this means that most probably, the deployment time of the Controller Node and its connection to each Worker is the operation with a bigger footprint in the deployment of the *k8s-osm-cluster-ns* NS. As for the scale-in operation, our results show an average of approximately 38 seconds. This big difference between both scaling operations can be explained by the fact that the scale-out comes with the execution of a Day 1 script to install all the Kubernetes components, beyond the creation of the Worker VM, while the scale-in only requires the destruction of that VM, without the installation of any tool.

VI. CONCLUSION

We proposed a MEC-NFV architecture for a B2B2C model, which allows a NSP to provide a full operational MEC infrastructure, with an independent management and orchestration platform, to each of its customers, the CSPs, in a MECaaS manner. Alongside this proposal, we also presented the needed modifications for a generic framework of a B2C model.

The tests we have conducted using this implementation allowed us to conclude that a NSP can deploy and provide to a CSP a fully functional MEC infrastructure in about 20 minutes. After the initial infrastructure is deployed, the tests show that MEC Applications can be launched with little to no extra infrastructure overhead. Furthermore, if a CSP requires more resources on its Edge environment, it only takes an average of 8 minutes to scale the number of Kubernetes Worker Nodes within the MEC host.

There are still some issues to be further studied in future works, such as the feasibility of having all the MEC functional blocks deployed under the same VNF, the placement and migration of MEC Applications, to better serve the corresponding UEs, or test this MEC architecture in a real Edge environment. However, with this work, we demonstrated the feasibility of a MECaaS system, as well as the usage of CNFs in MEC-NFV architecture and the deployment of the original MEC functional blocks as VNFs using the NFV infrastructure which an Operator might already possess.

ACKNOWLEDGMENT

This work is supported by the European Regional Development Fund (FEDER), through the Regional Operational Programme of Lisbon (POR LISBOA 2020) and the Competitiveness and Internationalization Operational Programme (COMPETE 2020) of the Portugal 2020 framework [Project NET-EDGE with Nr. 069977 (POCI-01-0247-FEDER-069977)].

REFERENCES

[1] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: State-of-the-art and research challenges," *Journal of Internet Services and Applications*, vol. 1, no. 1, pp. 7–18, 2010.

[2] ETSI, "Mobile-Edge Computing - Introductory Technical White Paper," Tech. Rep., 2014.

[3] ETSI, "ETSI GS MEC 003 V2.2.1," Tech. Rep., 2022.

[4] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Communications Surveys and Tutorials*, vol. 18, no. 1, pp. 236–262, 2016.

[5] V. Sciancalepore, F. Giust, K. Samdanis, and Z. Yousaf, "A double-tier MEC-NFV architecture: Design and optimisation," in *2016 IEEE Conference on Standards for Communications and Networking (CSCN)*, 2016, pp. 1–6.

[6] P. Sharma, L. Chaufournier, P. Shenoy, and Y. C. Tay, "Containers and Virtual Machines at Scale: A Comparative Study," in *Proceedings of the 17th International Middleware Conference*, ser. Middleware '16, Trento, Italy: Association for Computing Machinery, 2016.

[7] H. Abdah, J. P. Barraca, and R. L. Aguiar, "Qos-aware service continuity in the virtualized edge," *IEEE Access*, vol. 7, pp. 51 570–51 588, 2019.

[8] ETSI, "ETSI GS NFV-EVE 004 V1.1.1," Tech. Rep., 2016.

[9] ETSI, "ETSI GR NFV-IFA 029 V3.3.1," Tech. Rep., 2019.

[10] ETSI, "ETSI GR NFV-MAN 001 V1.2.1," Tech. Rep., 2021.

[11] G. Baldoni, P. Cruschelli, M. Paolino, *et al.*, "Edge Computing Enhancements in an NFV-based Ecosystem for 5G Neutral Hosts," in *2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, 2018, pp. 1–5.

[12] G. Cattaneo, F. Giust, C. Meani, D. Munaretto, and P. Paglierani, "Deploying CPU-intensive applications on MEC in NFV systems: The immersive video use case," *Computers*, vol. 7, no. 4, 2018.

[13] G. A. Carella, M. Pauls, T. Magedanz, M. Cilloni, P. Bellavista, and L. Foschini, "Prototyping NFV-based Multi-access Edge Computing in 5G ready Networks with Open Baton," in *2017 IEEE Conference on Network Softwarization (NetSoft)*, 2017, pp. 1–4.

[14] P. Fondo-Ferreiro, A. Estévez-Caldas, R. Pérez-Vaz, *et al.*, "Seamless Multi-Access Edge Computing Application Handover Experiments," in *2021 IEEE 22nd International Conference on High Performance Switching and Routing (HPSR)*, 2021, pp. 1–6.

[15] ETSI, "ETSI GS MEC 011 V2.2.1," Tech. Rep., 2020.