Performance Study of Multi-access Edge Computing Deployment in a Virtualized Environment

Supriya Dilip Tambe
Department of CSE
IIT Hyderabad, India
cs18resch11002@iith.ac.in

Yogesh Mandge
Department of CSE
IIT Hyderabad, India
cs18mtech11031@iith.ac.in

Antony Franklin A.
Department of CSE
IIT Hyderabad, India
antony.franklin@iith.ac.in

***Abstract*—Various real-time, latency-sensitive, and high-speed mobile applications are evolving as 5G applications. These applications are realized using Multi-access Edge Computing (MEC) and Network Function Virtualization (NFV) technologies in the 5G system. MEC platform, MEC service, and MEC application are the main components of an MEC Framework. NFV Orchestrator, Virtualized Infrastructure Manager (VIM), and virtualization technologies such as Virtual Machines (VM) and Containers are the main pillars of the NFV technology. In this paper, we study the impact of the virtualization technologies in the deployment of the MEC framework and its components while. We also study the impact of virtualization technologies on NFV and MEC KPIs such as onboarding time, instantiation time, MEC service and application response times. The experiments and its analysis show that containers perform better than VM to instantiate/terminate MEC components in the NFV framework. The observed MEC service KPIs show that the edge application's performance will be improved to meet the QoE of the applications irrespective of the virtualization technology used. These results can be used as a reference while deploying the MEC components based on their granular functionalities.**

***Index Terms*—NFV, MEC Framework, Virtualization Technologies, MANO, OpenStack**

I. Introduction

In telecommunication, 5G is the upcoming generation of mobile networks, providing enhanced capabilities in massive device connectivity. The main features of 5G are higher data rate, lower application latency, support for high mobility, higher spectral efficiency, and massive user connections [1]. Due to these features, various real-time applications can be deployed in the 5G network to serve a large number of users. The emerging application trends in the current mobile technology are High Definition video streaming, time-critical real-time communications in autonomous vehicles, Augmented Reality/Virtual Reality applications, and video gaming [2]. These applications require fast processing and low latency. Hence, next-generation cellular networks require advancements in latency, massive connectivity, local processing, scalability, and access to time-critical services to meet infrastructure demands and reduce back-haul congestion [16]. These real-time low latency services can be deployed in a 5G network with the help of the Multi-access Edge Computing (MEC) framework. MEC extends the computing functionalities of cloud computing by deploying services near the users and improves the

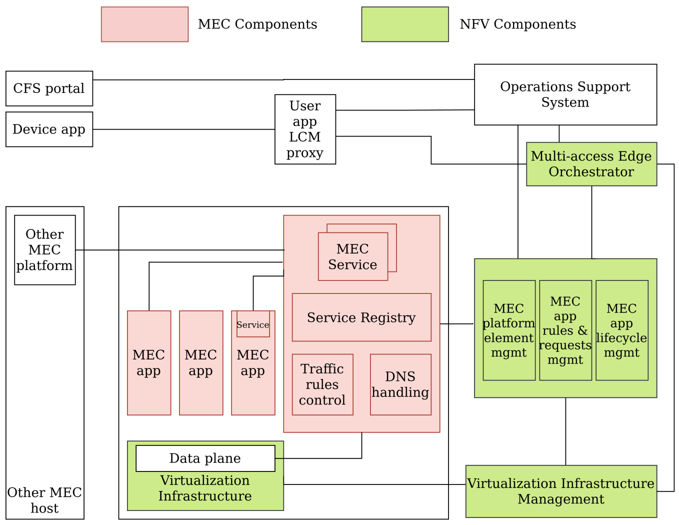


Fig. (1) MEC Deployment in the NFV Framework.

Quality of Experience (QoE). European Telecommunications Standards Institute (ETSI) provides the MEC Framework guidelines and its deployment options in the 5G network [3]. According to the ETSI MEC Framework specifications, the necessary components of a MEC Framework are MEC platform, MEC service, and MEC applications [4]. The MEC platform is responsible for handling the service registry, handling DNS, and traffic rules of the MEC application's data plane.

MEC services are the standard utilities required by the MEC platform and applications. Examples of the MEC services are Location service, Radio Network Information Service (RNIS), and 5GCoreConnect service. These services can be used to fetch information from the mobile network. Whereas MEC applications are real-time and low-latency server applications deployed at the proximity of users.

Network Function Virtualization (NFV) is used to deploy the MEC components. The benefits of using NFV are scalability, loose coupling, and distributed deployment of Virtual Network Functions (VNF). ETSI standardizes the NFV framework and also provides guidelines for MEC deployment in the NFV framework mentioned in [5]. NFV Orchestrator (NFVO), Virtual Network Function Manager (VNFM), and Virtual Infrastructure Manager (VIM) are the functional

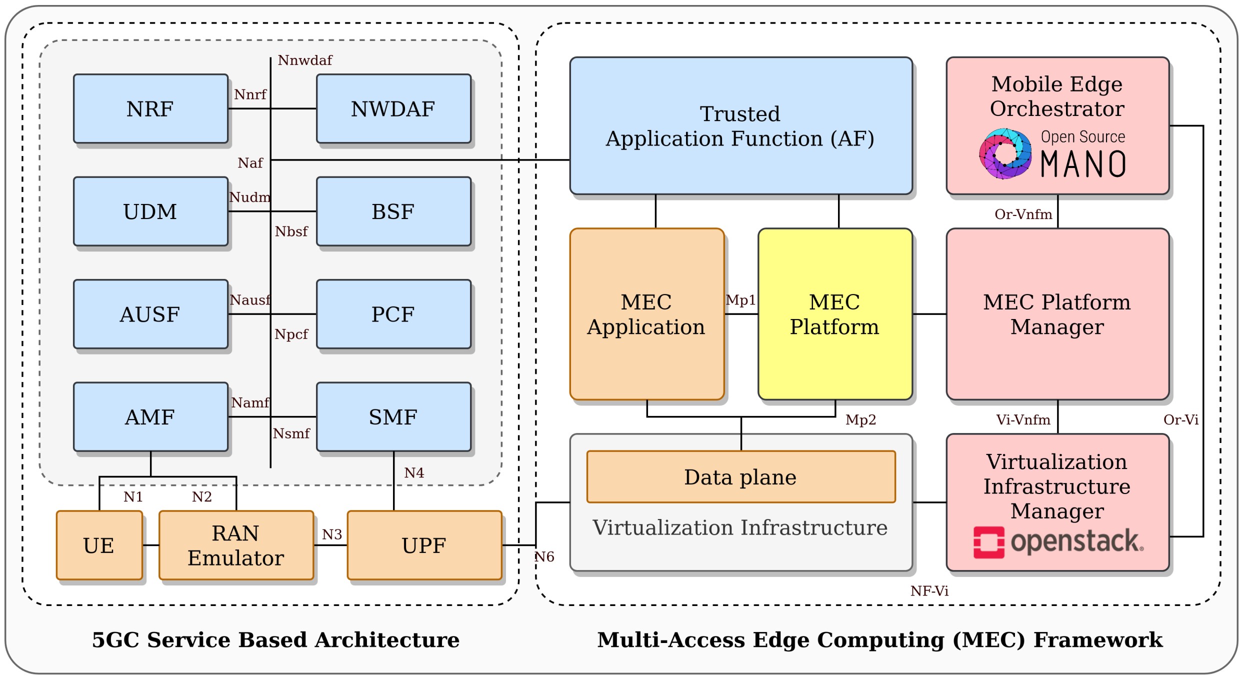


Fig. (2) Experimental Setup.

blocks of the NFV Management and Orchestration (MANO) framework. The NFVO is the central repository, which keeps records of VNF packages' instances, Network Service (NS) packages. The VNFM hosts MEC components and handles the lifecycle of VNFs. These VNFs are deployed using virtualization technologies such as Virtual Machines (VM) and Docker containers. VIM provides and manages the physical resources such as compute, storage, and network to deploy VNFs. Using NFV MANO, applications can be deployed in an ad-hoc and distributed manner. Fig. 1 shows the integration of MEC components in the NFV framework.

There are several open-source implementations of NFVO, such as ETSI OSM [6], and OpenBaton [7]. OpenStack [9], and OpenVIM [10] are some of the open-source implementation of VIMs. VMs and Docker containers are the leading virtualization technologies used to deploy VNFs. Whereas, Docker Swarm and Kubernetes [12] are used to orchestrate Docker containers. VMs are the primary component used in the cloud computing paradigm for the deployment of VNFs. As VMs come with their own Operating System (OS), it can be installed on bare metal or OS with a hypervisor's help. Due to the presence of guest OS, the deployment time and size of VMs is usually large. Docker containers are used as an alternative to VMs. The Docker container images contain only the libraries that are not part of the host OS, leading to small in size, portable, and faster deployment than the VMs. However, due to the presence of guest OS in VMs, it provides full isolation between running processes and secure processes compared to Docker containers.

In this paper, we aim to analyze the impact of virtualization technologies on the deployment of the MEC framework. We deploy and analyze the performance of MEC components on two popular VIMs, i.e., OpenStack and Vim-emu [11]. The rest of the paper is organized as follows, Section II describes the work done in performance evaluation of NFV, 5G, and MEC. Section III provides the details about the experimental setup, and Section IV describes the various experiments, results, and its detailed analysis. Section V concludes the paper with the future scope of this work.

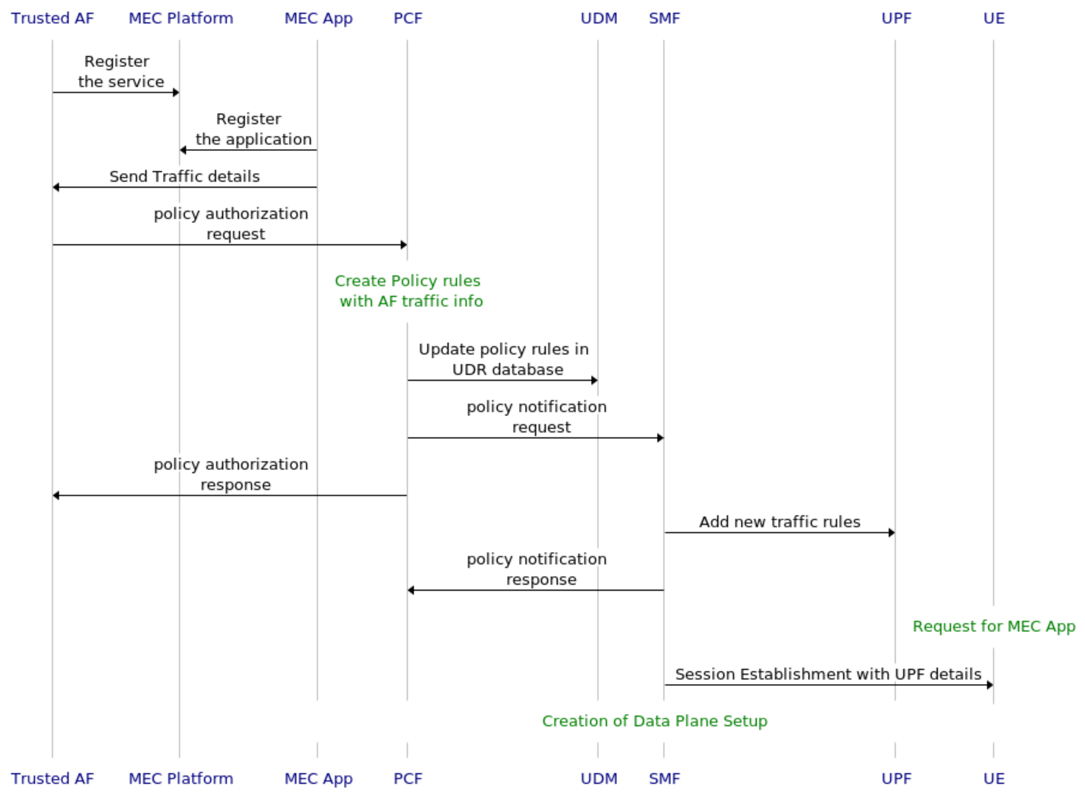
II. Related Work

A. 5G and MEC architecture

ETSI defined a reference architecture for MEC [3], which details the interaction between the MEC platform and the 5G network. It also describes various MEC deployment scenarios. [13] summarizes the various MEC deployment options in both 4G and 5G mobile networks, with the advancements in 5G. From the performance viewpoint, ETSI specifies MEC metrics and guidelines for MEC applications [14]. It covers significance and methods to collect values of various parameters such as Latency, Throughput, Energy efficiency, Resource footprint, and Quality of Service. In [16], the authors summarize the quantitative comparison between VMs and containers, the evolution of MEC, MEC deployment options, and comparison between different NFVOs.

B. NFV Components

The authors in [17] focused on the performance comparison of the NFVOs, OSM, and ONAP with functional and operational KPIs. The functional KPIs considered are the resource requirement to deploy the Orchestrator and the number of VIMs supported while the operational KPIs are the on-boarding time and deployment time. The authors of [18]



Trusted AF
MEC Platform
MEC App
PCF
UDM
SMF
UPF
UE
Register the service
Register the application
Send Traffic details
policy authorization request
Create Policy rules with AF traffic info
Update policy rules in UDR database
policy notification request
policy authorization response
Add new traffic rules
policy notification response
Request for MEC App
Session Establishment with UPF details
Creation of Data Plane Setup

Fig. (3) End-to-end service establishment.

covered the comparison of nine different NFVOs based on the available resource orchestration options and components in the NFV framework. The authors compared OpenStack and OpenVIM using CloudStack benchmarking tool in [19]. Performance metrics considered are VIM architecture, VIM resource requirements, VM image creation time, VM provisioning time based on VM size. The detailed comparison of virtualization technologies such as VM, container, and unikernel is covered in [20]. The authors studied various performance metrics such as request serving rate, average processing delay, CPU utilization, and memory utilization by varying number of requests and request size.

C. *MEC application deployment in NFV Framework*

The authors in [21] focused on CPU-intensive VNF deployment, such as a video caching application in the NFV framework. OpenBaton is used as the NFVO, and GPU based VNF deployed to evaluate its performance. Different aspects and issues in the deployment of edge robotics applications are covered in [23]. The authors also discussed the communication issues, workflow issues, and conceptual issues that are useful for validating any use-case deployment in an NFV framework. Deployment of high-quality 4K video streaming applications supporting low-latency in the MANO architecture is described in [22]. NFV-based 4K streaming MEC application and fuzzy moving DASH algorithms are compared for the latency and video segment arrival rates in [22].

In the literature, most work is focused on the general architecture and performance comparison between different NFV components, deployment of MEC applications in the NFV framework. To the best of our knowledge, there is not much work on deploying various MEC components in the NFV framework along with a 5G core network, which plays a vital role in deploying any real-time low-latency applications. Because the control plane communication is performed via these components, the placement and selection of virtualization technology are essential in the MEC framework deployment. Our primary motivation is to identify the research gaps in the deployment of MEC components in an NFV framework, which is done by using a real-time 5G experimental system, as described in the following sections.

III. Description of Experimental Setup

To study the interaction between 5G core entities and MEC components, we developed a 3GPP compliant proprietary implementation of 5GC as shown in Fig. 2. The 5G CN functions are developed using an HTTP/2 based Service-based architecture (SBA) and deployed as a Docker container. Also, to study virtualization technologies' behavior, we deployed the MEC components in the NFV framework using Openstack and vim-emu [11] as a VIM. Application Function (AF), which plays an intermediary role in the communication between 5G core and MEC Platform, is also developed using HTTP/2 libraries. AF also acts as a registered service in the MEC Framework. The MEC platform is developed in HTTP/1.1

using the OpenAPI interface descriptions provided by ETSI Forge [24]. We used real-time DASH based video caching application as our MEC application [25]. The application is divided into two modules, i.e., edge and cloud. The edge module is present at the MEC host, and the cloud module is hosted on the remote central server. Whenever a user requests for video, the request goes to the edge module. If the requested video is available, it returns the video frames, but the edge module sends the request to the cloud server if it is not available. Video frames are then delivered to the user while simultaneously being cached into the edge module's storage.

We have used ETSI OSM (Rel. 6) as our Mobile Edge Orchestrator Orchestrator (MEAO) and MEC Platform Manager (MEPM), which is compliant with the ETSI NFV guidelines. We have considered two VIMs, OpenStack 'Rocky' version and Vim-emu. OpenStack is used to deploy VNFs with VMs, and Vim-emu is used to deploy VNFs in the form of Containers. Docker swarm is used to connect MEC and 5G components. Vim-emu is chosen over Kubernetes for the deployment of Containerized Network Functions (CNF). At the time of this work, the integration of Kubernetes with OSM was new and had not matured. The system specifications used for the deployment of OpenStack and OSM are mentioned in table I.

TABLE (I) System Requirements.

	OSM	OpenStack
CPU	16 Cores	16 Cores
Memory	16 GB	16 GB
Storage	215 GB	215 GB

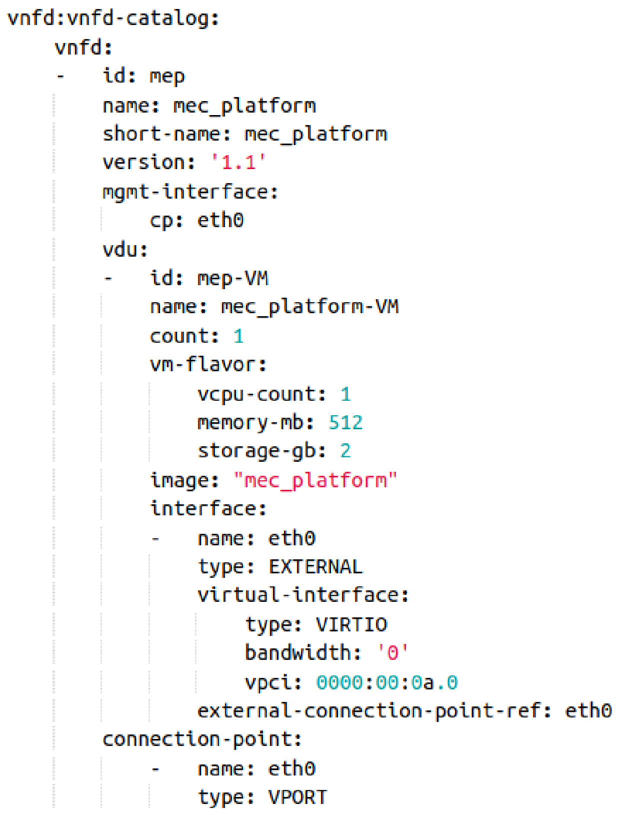
The call flow for end-to-end service establishment with the MEC application and 5G is shown in Fig. 3. Whenever the MEAO instantiates a MEC application, it registers in the MEC platforms service registry. The application's traffic routing details are then sent to the AF, which then forwards these details to the PCF. Upon receiving the routing details from the AF, PCF generates the corresponding PCC rules and sends them to the SMF, which adds these new routing details in the corresponding UPF. During UE session establishment with the 5G core network, SMF forwards the UPF details to UE. Then onwards, UE can directly access the services from the MEC application.

IV. Experimental Evaluation of the NFV Functionalities in a MEC Framework

This section evaluates the various KPIs of NFV and MEC, such as instantiation/termination time and service response time. It analyzes the impact of different virtualization technologies on the deployment of MEC components.

A. Application/VNF Onboarding Procedure

In the MEC Framework, the functional entities involved in the application package onboarding are the NFV Orchestrator and the VIM. The onboarding process is independent of the selection of VIM. Onboarding of any component of the MEC framework has the following steps:



```
vnfd:vnfd-catalog:
    vnfd:
    -   id: mep
        name: mec_platform
        short-name: mec_platform
        version: '1.1'
        mgmt-interface:
            cp: eth0
        vdu:
        -   id: mep-VM
            name: mec_platform-VM
            count: 1
            vm-flavor:
                vcpu-count: 1
                memory-mb: 512
                storage-gb: 2
            image: "mec_platform"
            interface:
            -   name: eth0
                type: EXTERNAL
                virtual-interface:
                    type: VIRTIO
                    bandwidth: '0'
                    vpci: 0000:00:0a.0
                external-connection-point-ref: eth0
        connection-point:
            -   name: eth0
                type: VPORT
```

Fig. (4) Example VNF Descriptor.

- Creation of VIM accounts for OpenStack and vim-emu.
- Creation of MEC platform images suitable to the VIM, i.e., Docker image for Vim-emu and QCOW2 image for OpenStack, respectively, and upload it on VIMs.
- Creation of Virtual Network Function Descriptor (VNFDs) and NSDs to specify the image name, resource requirement, networking, and initial configuration.
- Onboarding of the VNF and NS packages in OSM. This step does not require to specify VIM details.
- Selection of appropriate VIM account at the time of NS instantiation at the NFV Orchestrator.

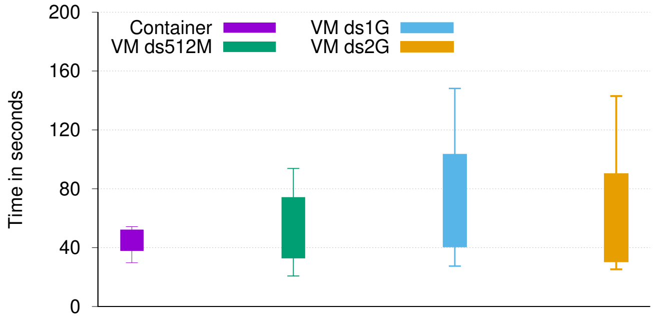
An example of VNFD to deploy the MEC platform in our experimental setup is shown in Fig. 4.

B. Network Service Deployment Time

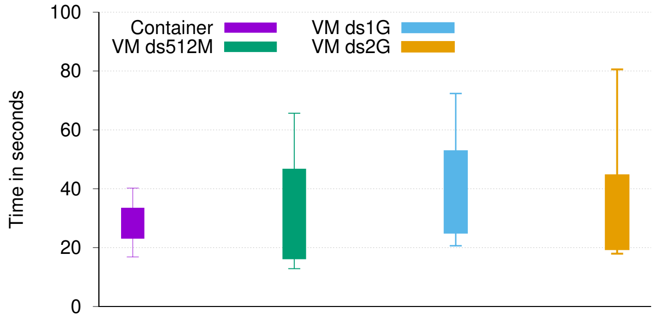
To analyze the performance impact of virtualization technology on NFV management entities during service deployment, we consider the following metrics,

- The time required to instantiate or terminate the MEC Platform from NFV Orchestrator.
- The time required to instantiate or terminate the MEC application from NFV Orchestrator.

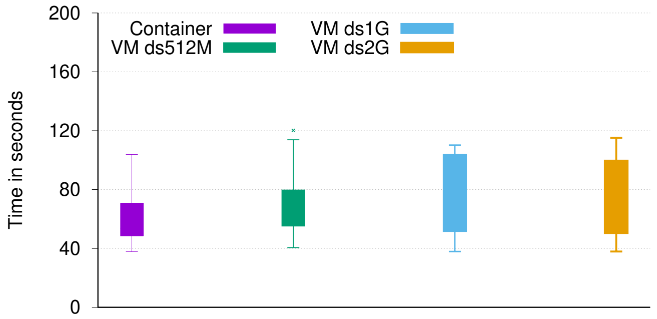
We created docker images and VM (QCOW2) images of the MEC platform, AF, and MEC application to realize it. The Docker image size for the MEC platform and AF is 1.2GB, and the MEC application is 700MB. The measurements were taken ten times on average with a varying load on the VIM and the



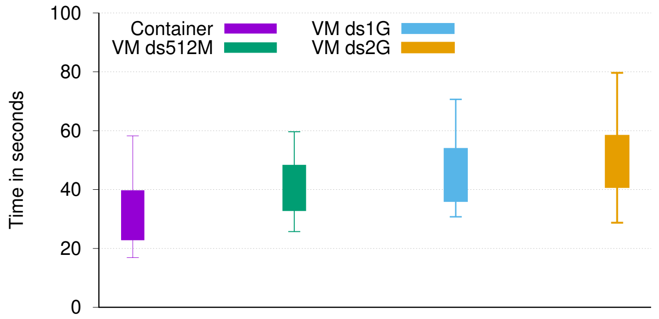
(a) MEC Platform Instantiation.



(b) MEC Platform Termination.



(c) MEC Application Instantiation.



(d) MEC Application Termination.

Fig. (5) Impact of Virtualization Technologies on MANO.

NFV Orchestrator. To make a fair comparison between VMs and Containers, the VM image's size was also considered. The details of different flavors used in the experiment are mentioned in Table II. MEC platform and AF are deployed as a single NS, Fig. 5.(a) shows the time required to instantiate the MEC platform and AF and register the AF service into the platform's service registry. Fig. 5.(b) shows the time required to de-register the AF service, terminate and cleanup of the MEC platform, and AF. Similarly, Fig. 5.(c) and Fig. 5.(d) shows the time required to instantiate and terminate the MEC application's NS.

TABLE (II) VM Flavors.

	ds512M	ds1G	ds2G
CPU	1	1	2
Memory	512 MB	1 GB	2 GB
Storage	5 GB	10 GB	10 GB

It can be seen from Fig. 5.(a) that time required to deploy the MEC platform on containers is less when compared to VMs. Furthermore, we can also see that most of the values are in a small range compared to VMs. Similarly, Fig. 5.(b) shows the time required for terminating the NS is less for containers than VMs. Fig. 5.(c) and Fig. 5.(d) show a similar trend for the deployment of MEC applications. However, there is a higher variation in the measured values than the MEC platform and AF. It can also be noted that the time required in all the cases is affected by the VM flavor, i.e., the larger the VM image size, the more time is required to instantiate/terminate the NS.

From the results, it can be inferred that the Containers perform better than the VMs. Hence, it makes them a perfect choice to be used as Virtualization technology in the quick deployment of the MEC framework. However, Containers lack complete isolation between processes and, thus, the security breach makes them unfavorable. Alternatively, we can use multiple Virtualization technologies to realize the MEC framework, i.e., deploy the MEC platform on VMs and use Containers for MEC applications.

C. Impact on MEC Service Response Time

In this section, we analyze the impact of virtualization technologies on the MEC Platform services. We only analyze the mandatory services required for service management and MEC application support, i.e., service registry, traffic rules, and DNS rules. Table III shows the response time for these services when deployed on two different VIMs, i.e., Vim-emu and OpenStack. From the table, we observe that the response time for both VIMs does not vary significantly. Because of the use of the HTTP request-response model, i.e., the implementation of these services is abstracted from the Virtualization technology. Also, these services are not compute-intensive to any virtualization platforms. So irrespective of the underlying virtualization technology, MEC services have the same performance.

TABLE (III) Service Response Time.

Service	Method	Vim-emu	OpenStack
Service Registry	Get	0.122s	0.129s
	Post	0.113s	0.119
	Put	0.116s	0.124s
	Delete	0.127s	0.130s
Traffic Rules	Get	0.123s	0.128s
	Put	0.150s	0.159s
DNS Rules	Get	0.114s	0.127s
	Put	0.199s	0.204s

D. MEC Application Response Time

Using the experimental setup, we measured the application response time for the video caching application based on video availability at the edge. The video application's application response time is measured as the time to get the first chunk of the video by the DASH-supported browser. Each chunk contains a video frame for 2 seconds of playtime. We conducted this experiment using vim-emu as a VIM. If the video is not available at the edge site, the average response time as about 215 milliseconds as the video needs to fetch from the remote server. If video is available at the edge (cached at the MEC), the response time is around 86 milliseconds. This reduction in time duration would benefit the application significantly in terms of video QoE.

V. Conclusion

Support for MEC in 5G is one of the main features of 5G to meet the requirements of various services envisioned for 5G by deploying the applications near the end user. NFV helps the management and orchestration of the MEC applications in an efficient way to deploy the MEC service/application as a VNF. These VNFs are realized through various virtualization technologies, such as VM and containers. In this paper, we have developed a MEC framework integrated with a 5G core as a testbed system to study the MEC components' performance with different virtualization technologies. MEC platform and MEC application instantiation require between 21 and 93 seconds and between 38 and 120 seconds, respectively, in both the virtualization technologies. To terminate the network services of MEC platform and application requires between 13 to 65 seconds and between 17 and 59 seconds. The time required to process the MEC platform's services varies between 0.113 and 0.204 seconds. The results show that virtualization technologies have a significant impact on the service deployment in the NFV framework. However, Containers perform better than the VMs in terms of instantiation time and image size. As per features of each virtualization technologies, and use case of deployed VNFs, mapping differs. Onboarding time and service response time are independent of virtualization technologies. We plan to extend this work to study the impact of virtualization technologies on the deployment of compute-sensitive MEC applications.

Acknowledgment

This work was supported by the project "Indigenous 5G Testbed" funded by the Department of Telecommunications (DoT), Government of India.

References

[1] "ITU-T Recommendation Y.3101 : Requirements of the IMT-2020 network", 2018.
[2] "ETSI GS MEC 002: Multi-access Edge Computing (MEC); Phase 2: Use Cases and Requirements ", 2018.
[3] "MEC in 5G networks", ETSI White Paper No. 28, 2018.
[4] "ETSI GS MEC 003: "Multi-access Edge Computing (MEC); Framework and Reference Architecture", 2019.
[5] "Network Functions Virtualisation (NFV)", ETSI NFV ISG White Paper, 2017.
[6] OSM, "https://osm.etsi.org/".
[7] OpenBaton, "https://openbaton.github.io/".
[8] OPNFV, "https://www.opnfv.org/".
[9] OpenStack, "https://www.openstack.org/".
[10] OpenVIM, "https://github.com/nfvlabs/openvim".
[11] M. Peuster, H. Karl, and S. v. Rossem, "MeDICINE: Rapid Prototyping of Production-Ready Network Services in Multi-PoP Environments", IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), USA, pp. 148-153, 2016.
[12] Kubernetes, "https://kubernetes.io/".
[13] Antony F. A, S. Tambe, "Multi-access edge computing in cellular networks", Springer CSI Special Issue on SDN, 2020, pp. 85-92.
[14] "ETSI GS MEC-IEG 006: Mobile Edge Computing; Market Acceleration; MEC Metrics Best Practice and Guidelines", 2017.
[15] "ETSI GR MEC 017 : Deployment of Mobile Edge Computing in an NFV environment", 2018.
[16] Taleb T, Samdanis K, Mada B, Flinck H, Dutta S and, Sabella D., "On Multi-Access Edge Computing: A Survey of the Emerging 5G Network Edge Cloud Architecture and Orchestration", IEEE Communications Surveys and Tutorials, vol. 19, pp. 1657-1681, 2017.
[17] Yilma, G.M., Yousaf, F.Z., Sciancalepore, V. and Costa-Perez, X., "On the Challenges and KPIs for Benchmarking Open-Source NFV MANO Systems: OSM vs ONAP", In Press, 2019
[18] L. Mamushiane, A. A. Lysko, T. Mukute, J. Mwangama and Z. D. Toit, "Overview of 9 Open-Source Resource Orchestrating ETSI MANO Compliant Implementations: A Brief Survey," IEEE 2nd Wireless Africa Conference (WAC), Pretoria, South Africa, pp. 1-7, 2019.
[19] Sechkova, T., Paolino, M. and Raho, D.,"Virtualized infrastructure managers for edge computing: Openvim and openstack comparison", IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB), pp. 1-6, 2018.
[20] Behravesh, R., Coronado, E. and Riggio, R., "Performance Evaluation on Virtualization Technologies for NFV Deployment in 5G Networks ", IEEE Conference on Network Softwarization (NetSoft) pp. 24-29, 2019.
[21] Cattaneo, G., Giust, F., Meani, C., Munaretto, D. and Paglierani, P., "Deploying cpu-intensive applications on mec in nfv systems: The immersive video use case", MDPI Computers, 7(4), 2018.
[22] Van Ma, L., Nguyen, V.Q., Park, J. and Kim, J., "NFV-based mobile edge computing for lowering latency of 4K video streaming", IEEE Tenth International Conference on Ubiquitous and Future Networks (ICUFN) pp. 670-673, 2018.
[23] Antevski, K., Bernardos, C.J., Cominardi, L., de la Oliva, A. and Mourad, A., "On the integration of NFV and MEC technologies: architecture analysis and benefits for edge robotics" Elsevier Computer Networks, 2020.
[24] ETSI Forge, "https://forge.etsi.org/".
[25] S Kumar, Vineeth DS., Antony F. A, "Edge Assisted DASH Video Caching Mechanism for Multi-access Edge Computing", IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS), pp. 1-6, 2018.