NETEDGE Proxy: A Multi-access Edge Computing System Level Proposal

André Martins[1], Marina Albuquerque[2], Vinicius Ferreira[1], João Bastos[1], Nicolás Lori[1], António Costa[1], Helena López[1], and José C. Bacelar[1]

[1] Algoritmi Research Centre/LASI, University of Minho, Braga, Portugal
[2] BEE Engineering, Portugal
Corresponding author: a84347@alunos.uminho.pt

Keywords: Edge computing, Multi-access Edge Computing, ETSI MEC, cloud-based technologies, NETEDGE.

1 Introduction

Multi-access Edge Computing (MEC) [1] is a solution proposed by the European Telecommunications Standards Institute (ETSI), that brings cloud processing closer to end users, enhancing computational capacity and reducing latency. In MEC systems, edge servers connect to virtualization infrastructure, delivering resources and services to hosted apps and enabling communication with centralized data centers and other MEC systems.

The initial ETSI MEC architecture evolved to a Network Function Virtualization (NFV) based architecture variant [1], which allows instantiating MEC Applications and Virtual Network Functions (VNFs) coexist, sharing resources, the virtualized infrastructure, and reusing ETSI Open Source Management and Orchestration (OSM) to fulfill a part of the MEC management and orchestration tasks.

We focus on MEC system level, which is responsible for users entry points, triggering application instantiation/termination, and resource orchestration. While there are existing projects providing MEC components, to our knowledge, our NETEDGE project [3] is the first to offer a comprehensive system-level stack designed for research, fully ETSI compliant, open-source, and freely available [2,3]. Although other projects such as Intel Smart Edge Open [4], Edge Gallery [5], and OpenAirInterface [6] provide valuable contributions, they either lack ETSI compliance or focus mainly on simulation and emulation functionalities [7].

This paper presents NETEDGE Proxy, a MEC system-level proposal, that uses MEC in NFV reference architecture, deploy the system-level components as KNFs and use an OSM client to integrate its functionalities to OSM. The remainder of this paper is organized as follows. Section 2 presents NETEDGE Proxy and its components. Performance tests results are presented in Section 3. Finally, Section 4 presents the paper conclusions.

[3] https://netedge.pt

2 NETEDGE Proxy

The ETSI MEC in NFV system level incorporates elements such as User Application Life-cycle Management Proxy (UALCMP) and Customer Facing Service (CFS) Portal. The UALCMP facilitates User Equipment (UE) applications to request the instantiation, termination, and movement of MEC applications within and beyond the system. The CFS Portal, on the other hand, enables operators to select MEC applications and receive service-level information from the provisioned ones. These requests are forwarded to the Operations Support System (OSS) for evaluation. The MEC Application Orchestrator (MEAO) and Network Function Virtualization Orchestrator (NFVO) are responsible for managing the lifecycle of the network services that include the MEC application instances, treating each MEC application instance as a VNF instance.

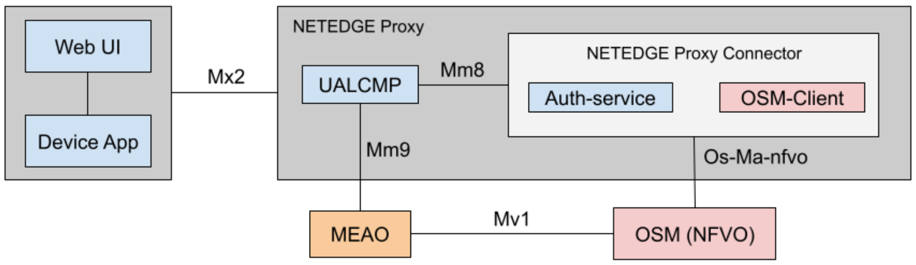

Fig. 1. Microservice architecture of MEC system-level components.

Figure 1 depicts our MEC System level proposal. We divide it in two parts, the client, with Web UI and Device App, and NETEDGE Proxy with UALCMP, NETEDGE Proxy Connector, OSM and MEAO.

The Web User Interface (Web UI) is a graphical platform where users interact with the system. It is a NodeJS-based graphical interface, and users must authenticate through OAuth 2.0 and OSM to access Edge services. After sign-in, users can operate apps tied to their account or related to their affiliated project(s).

The Device App acts as an interaction point for MEC services. It handles requests, including onboarding, instantiation, termination, and relocation of user apps within and beyond the MEC system. These requests are dispatched to the UALCMP via the standardized interface Mx2 [8], which subsequently forwards them to the appropriate endpoints—MEAO or OSM.

The NETEDGE Proxy, a crucial component, integrates essential UALCMP and OSS functionalities in the MEC NFV reference architecture. It facilitates communication between external entities (Device App and Web UI) and the orchestrators.

The UALCMP authenticates Device Apps' requests and connects with NETEDGE Proxy Connector and MEAO for further request processing. It uses the aforementioned Mx2 reference point to receive Device App requests. The Mm8 and Mm9 interfaces are non standardized. The NETEDGE Proxy Connector is a combination of the Auth-service and the OSM-Client[4]. The former authentic, verifies identities and provides permissions for MEC app instantiation or use. When users log in, they interact with the Auth-service, which utilizes the OSM-Client for user-associated project verification. Upon successful authentication, users receive an access token for future requests. The OSM-Client acts as a communication bridge between Auth-service and OSM.

The MEAO oversees and coordinates the MEC apps' deployment, lifecycle, and operation across the edge computing landscape. It interacts with other MEC components (e.g., UALCMP, OSS) to orchestrate MEC App operations. Though still under development, its current role is forwarding requests to the NFVO and ETSI MEC Host level entities. We use OSM as NFVO, and all the developed functional blocks are open source and available as a helm chart[5].

3 Results

We set up our test environment with a server and a laptop client. The server has two virtual machines (VMs) deployed on the VMware ESXi hypervisor. The first VM, MEC-OSM, was allocated 8 vCPUs and 12 GB of RAM, and it operated at CPU frequencies ranging from 400 MHz to 2.5 GHz. The second VM, MEC-POP0, was given 6 vCPUs and 12 GB of RAM. Each VM had approximately 130 GB of storage available.

MEC-OSM hosted the NETEDGE Proxy along with OSM and MEAO, emulating infrastructure management and access. MEC-POP0 represented the edge where user-requested MEC Apps were deployed. An external user is a laptop running the Web UI with the Device App. The client is connected to a Wi-Fi network that accesses the VMware server via local network.

We test the processing time and resources needed to process a client's request to deploy and terminate an application. We compare our tests with a request performed directly to OSM, as a ground rule. The tests are repeated 5 times and the results are averages and standard deviations.

Table 1 resumes the time spent on each step of a MEC Application deployment request. The Device App sends a request to NETEDGE Proxy, which processes, authorizes the request and forwards it to MEAO, that requests OSM to deploy the MEC App on MEC-POP0.

The ground rule here is the time spent on MEAO -> OSM -> MEAO communication, where the request is not processed by NETEDGE Proxy. The total time to process the requests using NETEDGE Proxy is 1.6599 ± 0.0841s for App deployment and 1.3129 ± 0.0380s for App termination. Therefore, the deployment took an additional 237.6 ms, and termination took an extra 136.4 ms on average compared to direct requests to the OSM.

[4] https://osm.etsi.org/gitlab/osm/osmclient
[5] https://github.com/UMinho-Netedge/NetEdge-charts

Table 1. Averege times and standard deviation at each stage of request

Deployment	Device App ->UALCMP	UACLMP ->MEAO	MEAO ->OSM ->MEAO
	0.0351 ± 0.0228 s	0.0913 ± 0.0209 s	1.4223 ± 0.0493 s
	Device App <- UALCMP	UACLMP <- MEAO	
	0.0215 ± 0.0268 s	0.0073 ± 0.0014 s	
Terminate	Device App ->UALCMP	UACLMP ->MEAO	MEAO ->OSM ->MEAO
	0.0094 ± 0.0032 s	0.0380 ± 0.0048 s	1.1765 ± 0.0251 s
	Device App <- UALCMP	UACLMP <- MEAO	
	0.0384 ± 0.0302 s	0.0086 ± 0.0016 s	

NETEDGE Proxy resources are deployed as containers and managed by Kubernetes. To measure the resources used by those resources we use Kubernetes metrics server, where the CPU usage by a container is denoted in milliCPU units and RAM usage is represented in Mebibytes (MiB), where 1 MiB approximates 1.0486 megabytes (MB). The UALCMP uses 5 mCore and 60 MiB, while NETEDGE Proxy Connector uses 4 mCore and 102 MiB.

Those test results reveal that although using NETEDGE Proxy slightly increases overall processing time and consumed computational resources, the benefits outweigh those costs. The benefits are the inclusion of an authorization service and proxy that enhance user experience and enables the processing of client requests using a standardized interface.

4 Conclusion

This paper presents NETEDGE Proxy, a MEC system level proposal, including the development of a client for the system. The processing time aligns with related work [2, 3] that directly accesses the NFVO, delivering ETSI-specified essential functionalities.

While all ETSI-specified system-level functionalities, such as MEAO, are yet to be fully implemented, this study serves as a stepping stone, available to the research community. Researchers can leverage this groundwork to further explore and develop the MEC system.

Acknowledgments

This work has been supported by FCT – Fundação para a Ciência e Tecnologia within the R&D Units Project Scope: UIDB/00319/2020.

References

1. ETSI, "Multi-access edge computing (mec); framework and reference architecture," 2022. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/MEC/001_099/003/03.01.01_60/gs_MEC003v030101p.pdf

2. V. Ferreira, J. Bastos, A. Martins, P. J. Araújo, N. Lori, J. Faria, A. D. Costa, and H. F. López, "NETEDGE MEP: A CNF-Based Multi-Access Edge Computing Platform," in *2023 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 2023, pp. 1–6.
3. P. Escaleira, M. Mota, D. Gomes, J. P. Barraca, and R. L. Aguiar, "Multi-access edge computing as a service," in *18th International Conference on Network and Service Management (CNSM) (CNSM 2022)*, 2022.
4. "Intel Smart Edge Open," https://smart-edge-open.github.io/, accessed: 2023-06-05.
5. "Edge Gallery," https://www.edgegallery.org/en/, accessed: 2023-06-05.
6. "OAI MEC," https://gitlab.eurecom.fr/oai/orchestration/oai-mec, accessed: 2023-06-05.
7. A. Noferi, G. Nardini, G. Stea, and A. Virdis, "Rapid prototyping and performance evaluation of etsi mec-based applications." *Simulation Modelling Practice and Theory*, vol. 123, no. 102700, 2023.
8. ETSI, "Multi-access edge computing (mec); device application interface," 2020. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/MEC/001_099/016/02.02.01_60/gs_MEC016v020201p.pdf