A MECApp-aware Lifecycle Management Approach in 5G Edge-Cloud Deployments

A MECApp-aware Lifecycle Management Approach in 5G Edge-Cloud Deployments

Paolo Bellavista, Armir Bujari, Luca Foschini, Andrea Sabbioni, and Riccardo Venanzi
DISI - Dipartimento di Informatica Scienze e Ingegneria, University of Bologna
Viale Risorgimento, 2, 40136 Bologna, Italy
Email: {name.surname}@unibo.it

***Abstract*—The recent trend pushing towards reliance on edge computing, virtualization and programmatic 5G network control has sparked the development of a myriad of open-source resource management and orchestration projects for improved control and added flexibility, making up for a rich and complex ecosystem of frameworks and tools with varying degree of support for standardized features. In this technological panorama, the ETSI Multi-Access Edge Computing (MEC) standard proposes a conceptual reference architecture, standardizing edge integration, interoperability and application management in an extended 5G edge-core architecture. In this article, we propose an application-aware orchestration solution for edge-core distributed deployments, currently lacking support in state-of-the-art frameworks and tools. The proposal is built on an experimental and distributed deployment of the OpenAirInterface minimal MEC platform implementation, and relies on the Kubernetes Operator pattern, targeting the automatic MECApp lifecycle management. To validate our approach, we conduct a series of experiments, reporting key metrics of interest.**

***Index Terms*—Edge, Cloud, MEC, 5G, NGN, RIC, Orchestration, Kubernetes, Operator**

I. Introduction

Mobile networks have undergone significant transformations, revolutionizing how we connect, communicate, and access information. The fifth generation mobile networks (5G) advocates for a comprehensive redesign of the technology, integrating traditional networks with modern technologies such as Network Function Virtualization (NFV), Software Defined Networking (SDN), Edge/Cloud computing, etc. [1]. In this context, one key focus is the integration into the 5G architecture of virtualization and orchestration tools, enhanced to contemplate advanced Edge/Cloud deployment models, easing the management of mobile applications within the Cloud Continuum paradigm [2].

ETSI Multi-Access Edge Computing (MEC), supports edge computing for IT services, enabling service providers to seamlessly manage layers of converged compute-network-storage resources while ensuring unified access for the users [3]. Moving services from the cloud to the edge reduces latency, enhances Quality of Service (QoS), and contributes to alleviating backbone network pressure [4]. Integrating 5G with the MEC model is crucial for ultra-low latency URLLC use cases, enabling service proximity and transforming private 5G networks into IT service platforms. A network combining 5G and MEC offers internet features and IT services through MEC applications (MECApps) deployed on MEC Hosts across distributed geographical locations. For these reasons, the integration of 5G with this architectural model has attracted the interest of the scientific community, leading to various incremental proposals in recent years [5], [6].

On this front, several all-in-one deployment solutions have emerged, easing the management and integration of 5G networks and the ETSI-MEC model. These solutions have been proposed with the goal of supporting the deployment and runtime management of 5G edge resources in a 3GPP-compliant fashion. Notable efforts from the project Aether, Mosaic5G, Open Network Automation Platform (ONAP), OAI-Operation and Maintenance (OAM) project, just to name a few, aim to integrate virtualized Radio Access Network (RAN) deployments and various Core Network (CN) functions with edge/cloud computing services and tools [7]–[10]. OAI-OAM emerges as a pivotal project for providing orchestration, monitoring, and maintenance of OAI-RAN and OAI-CN, and it is gaining significant attention within the research community due to its pioneering efforts in integrating 5G networks with intelligent features, including the support for am ETSI compliant MEC implementation. Although OAI-OAM is still at an early development stage when compared to similar other 5G-CN projects, it represents an interesting playground for practitioners and researchers as it proposes one of the first concrete attempts at the integration between 5G networks and ETSI-MEC.

However, the current version of the OAI 5G-MEC platform has limitations that make it unsuitable for managing real deployments in dynamic settings. The current deployment model imposes coupling constraints on network functions, reducing core network flexibility and hindering component scaling and migration. Additionally, the lack of automation for edge application deploymentand the absence of mechanisms for the runtime relocation of applications between MEC host instances, limits the experimentation to static (edge) application scenarios. These limitations underscore the urgent need for a flexible service orchestration mechanism, allowing for the dynamic management of MECApps.

To this end, we propose a hierarchical layered orchestration solution that enables a MECApp-aware lifecycle management, moving the original and traditional OAI-OAM static deployment model towards a more dynamic and managed one. Our approach advocates for a hierarchical orchestration framework,

which relies on the Kubernetes Operator pattern [11], for monitoring and automating all the operations needed to distribute the application along the possible edges (MEC platforms) of the network according to operators' configurable policies. To validate the approach, we conduct a series of experiments in real distributed 5G-MEC deployment, dynamically triggering the deployment and removal of MECApps, and discussing the resulting performance trend.

The article is structured as follows: Sec.II provides background information on the platform being examined, emphasizing its shortcomings in managing resources during edge-core deployment scenarios. Section III outlines the proposed solution, which includes distributed deployment, a MEC Orchestrator, and a mobility management feature. Section IV presents test results comparing the performance of MECApp's mobility operations on two different clusters. Finally, Sec.V concludes the article.

II. Background and Motivation

There is a vast body of literature related to ETSI-MEC, NFV and orchestration, advocating for solutions to specific technical problems emerging at different layers of the technological stack [12]. To the best of our knowledge, there is a lack of studies devoted to the actual implementation and field evaluation of ETSI MEC deployments for the automatic MECApp configuration and management. In this section, we briefly survey the targeted software ecosystem, stating the desirables for a software stack targeting the seamless management of applications in edge-cloud deployments.

OAI-OAM offers a solution that integrates the classic 5G architecture with an edge application platform based on the ETSI-MEC standard. The technological stack of OAI-OAM is depicted in Fig.1 and it is composed of the following main components:

- **OAI-RAN & OAI-CN:** implementing the functional components of the 5G RAN and Core Network with the possibility of using a RAN simulator instead of physical radio cells.
- **FlexRIC:** implementing a Radio Intelligent Controller (RIC) compliant with the O-RAN specification. xApps could be developed thanks to the availability of a dedicated SDK; a default xApp (RNIS) is available and used to receive radio signal quality metrics, publishing them for use by OAI-RNIS.
- **OAI-CM:** enabling the monitoring (and in the future, controlling) the 5G Core Network. Currently, it is only possible to monitor the events exposed by OAI-CN's AMF and SMF.
- **OAI-MEP:** Minimal working example of a MEC Platform, hosting MECApps, designed following the guidelines of the MEC standard. On the management side, the support is currently limited to MECApp discovery and registration via the Mp1 interface.
- **OAI-RNIS:** Implementation of a MEC service that exposes metrics collected by RNIS xApp and events captured by OAI-CM to the operator.

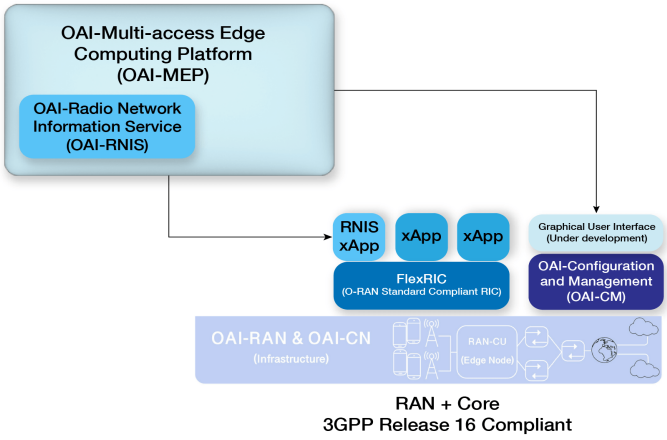

Fig. 1. OAI-OAM technological stack [10]

While the platform offers comparatively less support for 3GPP standardized features than similar initiatives, it represents an interesting playground for researchers and practitioners as it proposes one of the first attempts at concrete integration between 5G networks and a standardized MEC solution. Without loss of generality, OAI-OAM currently limits its support to a manual MEC application deployment. While the framework does provide some basic hooks for edge application management, e.g., the Mp1 ETSI-MEC interface, this capability alone does not suffice and confines the platform usage to illustrative and educational purposes. Moreover, the current deployment model poses several coupling constraints between the various network functions, limiting the support to purely static use-cases with a fixed deployment chosen *a-priori*. We argue that dynamicity is essential for edge application scenarios, hence there is a need for a new architecture that leverages flexible and elastic service orchestration mechanisms. Addressing the limitations, we introduce dedicated support for MECApp application management, leveraging existing signaling interfaces for application lifecycle management. Our proposal relies on a hierarchical orchestrator based on the Kubernetes Operator pattern and is comprised of a cloud orchestrator and a MEC one.

These loosely coupled operators coordinate to enact the migration of the applications/services from/to core/edge according to operators' pre-configured policies. Supporting inter-cluster application migration, paramount is the capability to preserve service continuity, hence the user' established session. In this direction, we discuss several potential solutions with varying degrees of support.

III. A Layered Orchestration Approach for MECApp Lifecycle Management

In this section, we present the extension to OAI-OAM, adding support for application management in edge-core deployments. Firstly, we start discussing a general deployment topology and component distribution. Next, we discuss the layered orchestration framework and the required minimal integration with the OAI platform currently supported.

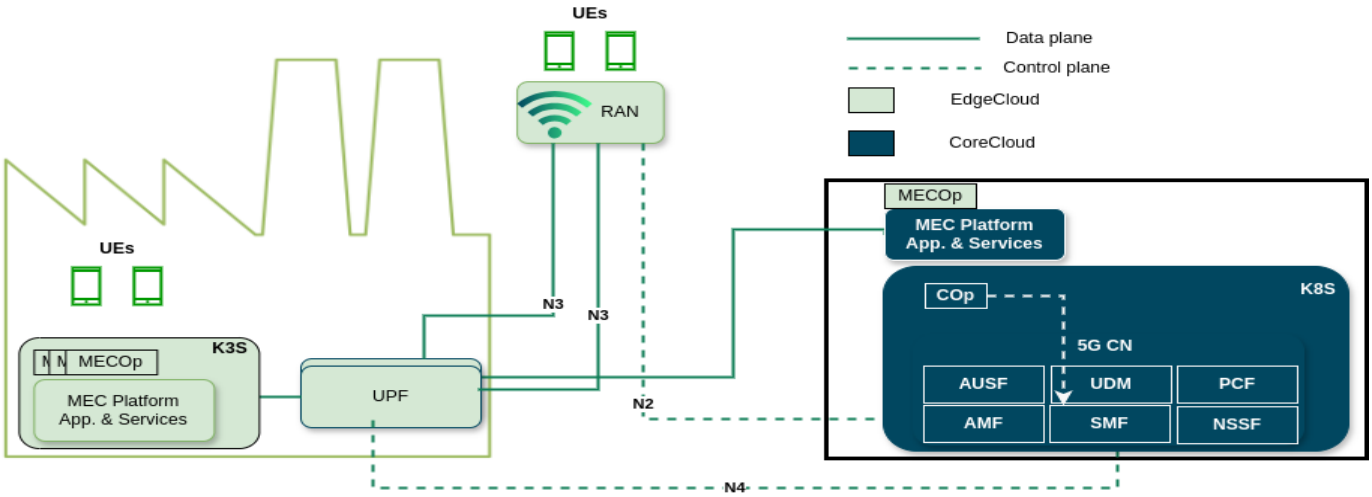

Fig. 2. OAI-OAM distributed deployment consisting of two MEC sites and the core network one. On the left-hand side is depicted a shopfloor environment (factory) with a cluster hosting MECApps serving low-latency, high-bandwidth and best-effort industrial applications. On the right-hand side is the CoreCloud deployment, hosting the 5G SBA and application services. The two sites are connected to each other via a site-to-site VPN.

A. Distributed Edge-Core 5G-MEC Deployment

The target distribution model comprises various MEC platforms (edges) alongside a cloud component responsible for hosting the 5G Service Based Architecture (5G-SBA). The cloud component is tasked, among other things, with the management and resource distribution among the edges. A simplified model is depicted in Fig. 2, serving as a research playground to assess intelligent resource management, with particular emphasis on QoS/QoE-aware management in mixed-criticality environments. On the left-hand side of Fig. 2 is depicted a factory (shopfloor) environment servicing heterogeneous (control) applications with different QoS specifications. In this setting, applications with stringent QoS requirements in terms of latency shall be serviced locally - MECApp installed in the local edge site shown in green - without incurring any additional delays required to traverse the public telco. domain. Conversely, best-effort applications can be deployed on the cloud side shown on the right-hand side.

On a practical note, the testbed comprises two different Kubernetes clusters, a more capable cluster running a vanilla Kubernetes (K8s), hereafter referred to as CoreCloud cluster, on which both the 5G-SBA and an instance of the MEC Platform are hosted, and another one, the EdgeCloud, deployed using a lightweight Kubernetes distribution (K3s), hosting another instance of MEC Platform. Having the MEC platform installed on the CoreCloud cluster is for economic purposes to assess potential cross-edge migration scenarios.

The 5G RAN, VPP-UPF(s), and the UE are colocated and installed on a bare metal machine hosted at the EdgeCloud but not subject to its dynamics. The 5G User Plane Function (UPF) would benefit from the flexibility that an orchestration tool like Kubernetes provides, however, its integration as a Kubernetes CNI is outside the scope of this work. The testbed is composed of five nodes: two high-end servers comprising the CoreCloud; two commodity nodes comprising the EdgeCloud, and a commodity PC hosting the UE, RAN, and two UPF instances (later on). Currently, the UE and RAN components are simulated, while the UPF is a software-based instance running in user-space, making use of acceleration techniques to achieve comparable performance to hardware-based switches.

In order to deploy OAI-OAM within our Kubernetes Clusters, it is necessary to specify the configuration of the various components through appropriate Kubernetes manifests. To this end, we need to transform the traditional Docker Compose configurations into Kubernetes manifests. Each manifest contains the definition of two Kubernetes objects: a Deployment, and a Service. The first manages and defines the Pods on which the containerized component will run, while the other describes the services that the component exposes to clients outside the cluster. For the sake of conciseness, we do not list all the steps and the network configurations needed to correctly setup the cluster.

B. MECApp Orchestrator and MECApps Management

To implement the MECApp lifecycle management we need to develop a specific component capable of automating: the distribution, the deployment, the deletion, the configuration, and the migration of MEC applications. Our proposal aims to implement a smart and seamless lifecycle management of services and applications across cloud and edge sites. Our approach relies on the Kubernetes Operator pattern and is composed of two components (Fig. 3): (i) the Cloud Operator which maintains a global view of the state of available resources and services deployed, and (ii) the MEC Operator(s) which implement local, specialized management of the deployed services.

Starting from the bottom of Fig. 3, one finds the MEC Operator (MECOp) which represents the first layer of ab-

straction, implementing the orchestration logic for a specific service or application hosted at the edge (MEC). The rationale behind adopting an application-specific orchestration flow is to cater to the applications' specific needs in terms of scaling, fault tolerance, data dependencies, etc. As an example attesting to this practice, the authors in [13] make use of the Operator pattern to scale massively parallel deep learning workloads across clusters, which is a very difficult task to achieve via the native Kubernetes scheduler. Other notable use cases, commonly adopting this orchestration pattern, fall within the class of stateful applications, whose scaling or fault recovery procedure, requires specific considerations on state management.

The MEC operator implements a control loop, which periodically checks the *etcd* database as a unique source of truth for the desired cluster state. Whenever the MECOp detects a deviation between the observed state and the retrieved desired state, it executes the custom management logic for state reconciliation. In order to simplify the management for the Cloud Operator(s), MEC Operator exposes a set of specific APIs providing support for operations like Create, Observe/Read, Update, and Delete (CRUD) for service management, implementing primitives for service creation, observability of structure/service state, update, and deletion from the edge site(s). CRUD operations directly modify service definitions stored on local MEC *etcd* deployments, successively triggering the COp (later) to enact an adaptation process, i.e., MECApp migration. As an example, metrics of interest are the applications' mean response time, memory consumption, maximum latency perceived etc. In particular, during the Observe phase, each specialized MECOp captures metrics evaluating the operational state of the application and the environment. If the application state is not compliant with the desired Service Level Objectives (SLOs), specified in the service configuration, the MECop marks the status of the deployment as *degraded*.

The Cloud Operator (COp) implements the second layer of abstraction and orchestration logic. It exposes a set of customized APIs, allowing end-users to define, submit, and delegate the management of a generic type of MEC service. The COp retrieves service definitions and enacts a control loop that embodies the logic for service lifecycle management, starting with a deployment phase on a specific MEC site fulfilling the requirements. When a new service description is submitted to an observed repository, i.e., Github, a dedicated CI/CD pipeline enacts the deployment process by submitting a configuration file describing the characteristics and requirements of the service

The definition of the new service is stored in the *etcd* registry, from which the COp fetches new configurations and potential changes to them. Upon retrieving the service definition and the actual infrastructure state - of the controlled cloud and edge resources via the C(R)UD interfaces - the COp decides the most advantageous location for hosting it. If the deployment cannot be enacted, such as faults or the required resources for service deployment are not available,

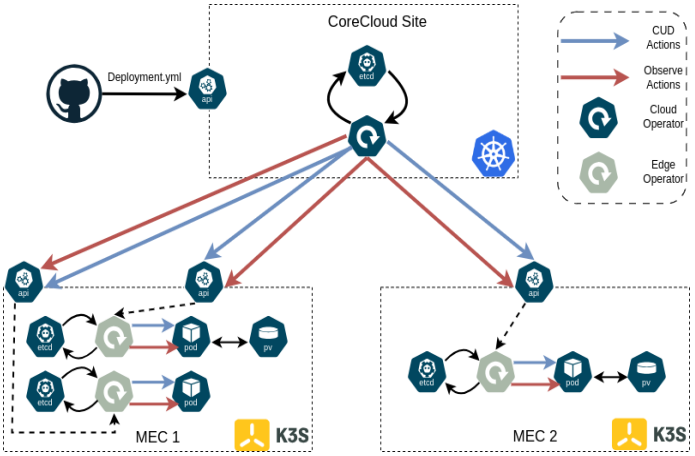

Fig. 3. Hierarchical Core-Edge Operator Functional Diagram

the COp defers the plan and periodically polls for the new status until the service requirements are satisfied. Otherwise, if the deployment in one of the edges is successful, the COp will at first update the status of the resource in *etcd*, marking the resource as deployed, and enacts an infinite loop that periodically polls the state of the deployed resource through the MOps APIs. In the evidence of sudden unavailability of the service or degraded performance, the COps will start an adaptation procedure, initiating the migration of the service from the original MEC site to another suitable one.

The abstractions implemented by this hierarchical orchestration framework simplify the management of applications across edge-cloud sites, enabling control plane interactions via intuitive ReST interfaces, which can easily be integrated and used by a higher-level business logic. This decoupling among the core and edge orchestrators, and their respective states, allows the MEC Operator to account for the specificities of each hosting environment, such as the availability of resources and technologies installed. Also, by separating the global and local states, our solution trades off strong consistency to achieve higher reliability and partition tolerance. This means that in the event of a failure, MOps and services hosted on MEC sites can continue to operate while awaiting reconnection to the cloud site.

C. Preserving Service Continuity

The feature is particularly relevant when latency-critical applications are serviced via 5G. To this end, several solutions are available, ranging from network-centric approaches, requiring no involvement from the UE, to solutions where the UE contributes to the mobility management scheme [14]. Other solutions in this spectrum include UE-centric approaches, where the UE relies solely on a DNS for (new)service discovery.

Contributing to extend the mobility management support in OAI, we make a leap forward by introducing a feature allowing for the dynamic reconfiguration of the data plane (UPF). The process involves the modification of the impacted (established) PDU session, and a reconfiguration step involving the UPF

and RAN with the end goal of routing packets to the appropriate MEC platform. Currently, neither the OAI-Policy Control Function (OAI-PCF) nor the OAI-SMF have built-in functionalities that allow the modification of pre-existing PDU sessions; the VPP-UPF, however, provides the necessary hooks. To implement this feature, we introduce a specific ReST endpoint to OAI-SMF to expose the reconfiguration service, adding a new SMF procedure implementing the reconfiguration logic enacted and triggered by the SMF via an explicit solicitation by the COp.

In the current implementation, the reconfiguration process enacted by the OAI-SMF makes the necessary adjustments to the SMF context, triggering the UPF (re)configuration via the N4 reference point. This message is succeded by another one sent via the N2 reference point, signaling the RAN to adopt a new path for routing the data packets. The approach, although not fully compliant with 3GPP (see below), works thanks to the availability of two UPF instances, each attached to a different interface.

It is noteworthy to point out, that the actual implementation of the reconfiguration process is not fully 3GPP compliant, short-circuiting some signaling that has to occur in the 5G-SBA for a correct propagation of the reconfiguration information, e.g., the process should also keep the AMF component, currently circumvented, informed on the enacted changes. The current solution, however, is a step toward a fully compliant solution, allowing practitioners to assess inter-cluster 5G-native MECApp migration mechanisms.

IV. Validation and Experimental Analysis

In this section, we assess and validate our proposal in the extended edge-core deployment model. The study represents the first step of a wider research objective, targeting the runtime stateful MECApp migration, involving also cross-domain cooperation [15].

A. Experimental Setup

The testbed employed for the experimentation is the one depicted in Fig. 2 and described in Sec.III-A. Adding to its description: the COp is deployed on the CoreCloud site and the MECOp is deployed in each MEC platform. In the first experiment, we are interested in validating the COp-MECOp interaction loop, triggering the dynamic deployment and deletion of a MECApp (COp) measuring the time required to complete parts of the process. In the second experiment, we demonstrate a dynamic scenario where the orchestration layer initiates a runtime MECApp migration in response to a violation of SLO specification.

B. Preliminary Results

To trigger the dynamic deployment and deletion of a MECApp, we setup a namespace *mecapp* on both clusters where the various MEC applications can be hosted. The dynamic deployment or removal of a MECApp, is triggered by the creation/deletion of a service definition configuration fetched by the COp. The creation process is composed of the deployment phase of a target MECApp and its registration to the OAI-MEC platform registry to make the service available to be consumed. On the other hand, the deletion process is composed of the deregistration of a MECApp from the registry and the total removal of the service from the node.

Figure 4 shows the breakdown of the runtimes for the creation and deletion processes on the K8s cluster and K3s cluster. The chart shows a pure quantitative result because the type and the size of the MECApp clearly influence the outcome. Herein, the image download and service instantiation times have been aggregated in MECApp Deployment time. This time could be additionally reduced by having edge sites rely on a local or distributed file system (repository) for image download [16]. Also, we experimentally found out that additional performance improvements can be achieved by writing the Operators' tasks in GO native language instead of Ansible as in the current implementation. In this way, the operators' tasks, especially the deployment and removal one, would perform up to four times faster [17]. Additionally, it is important to note, that the timing of both phases is also influenced by the COp polling interval, querying the *etcd* databased, currently set to *1 second*. On this front, we are working on an event-driven abstraction layer, to allow for timely notifications on deployment/configuration changes.

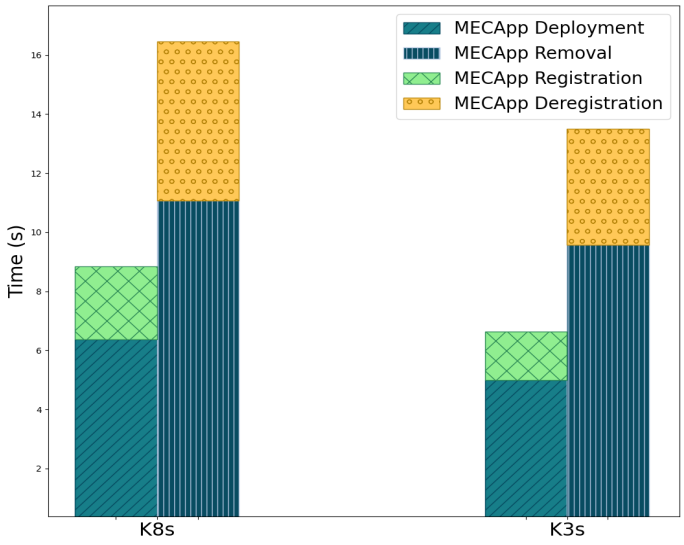

Fig. 4. MECApp creation and deletion times averaged over 50 runs of the experiment.

In the second experiment, we show the migration technique in action, triggered as a consequence of a rule (QoS) violation. In this scenario, we use a simple MECApp which echoes back a UE sent a message. This scenario aims to mimic an industrial control loop logic which should be completed within a predetermined time; data freshness is paramount and UDP is used at the transport layer. In this context, the UE represents an industrial machine equipped with a sensor and an actuator, sending operational data in the uplink, and receiving process control information from the edge application in the downlink. For simplicity, the roundtrip time is measured at the UE and

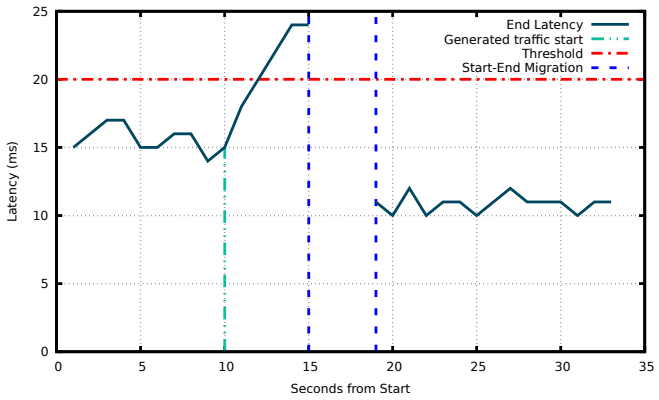

Fig. 5. Perceived latency. A sudden increase of RTT is perceived starting at time t=10s, degrading MECApp performance, and triggering the COp to initiate the migration process from the CoreCloud to the EdgeCloud MEP site.

announced out-of-bound to a MEC platform service, consulted by the COp. To obtain the RTT metric, one could also rely on the N4 QoS reports, but this integration is left as a future work. The MECApp is initially installed in the CoreCloud, right-hand cluster depicted in Fig. 2, having a baseline RTT of 15ms. The QoS specification of the MECApp requires the entire *control loop* to be completed within 15ms and not exceed the acceptable service threshold of 20ms.

Figure 5 shows the RTT evolution as reported by the UE. Until time t=10s the RTT is acceptable with small fluctuations attributed to cross traffic on the public (shared) link connecting the two sites. At time t=10s, we generate synthetic cross traffic at the ingress interface of the MEC platform at the CoreCloud deployment, contributing to a steady increase of the perceived RTT on the UE side. The COp eventually observes the persistent QoS violation, and at time t=15s, triggers the migration process of the MECApp - from the MEC platform at the CoreCloud site to the EdgeCloud - enacting also the data plane reconfiguration (SMF). Once the service migration procedure is completed, from time t=19s the MECApp becomes fully operational, servicing the UE within the acceptable QoS bounds.

Acknowledgments

This work is partially supported by the European Union - NextGenerationEU - National Recovery and Resilience Plan (Piano Nazionale di Ripresa e Resilienza, PNRR) - Project: "SoBigData.it - Strengthening the Italian RI for Social Mining and Big Data Analytics" - Prot. IR0000013. The authors thank Elisa Drudi for the initial support of the project activities.

V. Conclusion and Future Directions

Herein we presented a possible extension to the OAI-OAM platform which offloads component/application management to Kubernetes. The solution relies on the Kubernetes Operator pattern, implementing an application-aware lifecycle management across 5G edge-core deployments. The proposal was assessed and validated in a real deployment environment, consisting of a core and edge cloud, demonstrating the solutions' ability to enact the migration of application components from/to the edge/core and across edges. Supporting time-critical applications is a future work we aim to pursue. To this end, as future research direction, we plan to enhance the orchestration layer along these two main directions: (i) add support for an event-driven notification layer and (ii) deeper integration with CN-RAN signaling mechanisms implementing a minimal analytics function feeding the COp.

References

[1] F. Spinelli and V. Mancuso, "Toward Enabled Industrial Verticals in 5G: A Survey on MEC-Based Approaches to Provisioning and Flexibility," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 1, pp. 596–630, 2021.

[2] L. Bittencourt *et al.*, "The internet of things, fog and cloud continuum: Integration and challenges," *Internet of Things*, vol. 3-4, pp. 134–155, 2018.

[3] ETSI, "Multi-access Edge Computing (MEC); Framework and Reference Architecture," European Telecommunications Standards Institute, Technical Specification (TS), 2024, ver. 3.2.1. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/MEC/001_099/003/03.02.01_60/gs_MEC003v030201p.pdf

[4] ——, "Multi-access Edge Computing (MEC); Use Cases and Requirements," European Telecommunications Standards Institute, Technical Specification (TS), 2024, ver. 3.2.1. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/MEC/001_099/002/03.02.01_60/gs_MEC002v030201p.pdf

[5] 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, p. 102700, 2023.

[6] F. Asquini, A. Bujari, D. Munaretto, C. E. Palazzi, and D. Ronzani, "An ETSI NFV Implementation for Automatic Deployment and Configuration of a Virtualized Mobile Core Network," in *Proc. of IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, 2021, pp. 357–362.

[7] Open Networking Foundation. (2024) Aether Private 5G Project. [Online]. Available: https://docs.aetherproject.org/master/intro.html

[8] EURECOM. (2024) Mosaic5G Data-driven Service Delivery Platform. [Online]. Available: https://gitlab.eurecom.fr/mosaic5g

[9] ONAP Linux Foundation. (2024) Open Network Automation Platform (ONAP). [Online]. Available: https://docs.onap.org/en/kohn/platform/overview/index.html

[10] Open Air Interface. (2024) OAI Operations and Maintenance Group. [Online]. Available: https://openairinterface.org/projects/oam-project-group

[11] J. Dobies and J. Wood, *Kubernetes operators: Automating the Container Orchestration Platform*. O'Reilly Media, 2020.

[12] A. Filali, A. Abouaomar, S. Cherkaoui, A. Kobbane, and M. Guizani, "Multi-access edge computing: A survey," *IEEE Access*, vol. 8, pp. 197 017–197 046, 2020.

[13] A. Kanso *et al.*, "Designing a Kubernetes Operator for Machine Learning Applications," ser. WoC '21. Association for Computing Machinery, 2021.

[14] "IP Mobility Support for IPv4," RFC 3344, 2002. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc3344

[15] ETSI, "Multi-access Edge Computing (MEC); Federation enablement APIs ," European Telecommunications Standards Institute, Technical Specification (TS), 2024, ver. 3.2.1. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/MEC/001_099/040/03.02.01_60/gs_MEC040v030201p.pdf

[16] R. A. Addad, D. L. Cadette Dutra, M. Bagaa, T. Taleb, and H. Flinck, "Towards a Fast Service Migration in 5G," in *Proc. of IEEE Conference on Standards for Communications and Networking*, 2018, pp. 1–6.

[17] J. Geerling. (2021) Fast vs Easy: Benchmarking Ansible Operators for Kubernetes. [Online]. Available: https://www.redhat.com/en/blog/fast-vs-easy-benchmarking-ansible-operators-for-kubernetes