MEC Application Migration by Using AdvantEDGE

Prachi Vinod Wadatkar[1,2(✉)], Rosario G. Garroppo[2], and Gianfranco Nencioni[1]

[1] Department of Electrical Engineering and Computer Science, University of Stavanger, Kjell Arholms gate 41, 4021 Stavanger, Norway
{prachi.v.wadatkar,gianfranco.nencioni}@uis.no
[2] Department of Information Engineering, University of Pisa, Via Girolamo Caruso, 16, 56122 Pisa, Italy
rosario.garroppo@unipi.it

Abstract. Multi-access Edge Computing (MEC) and 5G are key technologies for the development of new applications requiring low latency and for computation off-loading. Emulation tools, such as AdvantEDGE, allow to rapidly test new services and resource management techniques in the 5G-MEC infrastructure. The paper presents an experimental study aimed to show the usage of AdvantEDGE tool for evaluating the migration performance of a MEC application. The key aspect of the study is that the application mobility is obtained by using the migration of the Kubernetes (K8s) application pod. The standard K8s does not have the ability to support the pod migration in a cluster of nodes. While recent research proposes a mechanism to migrate pod, there is no work investigating the migration technique with the AdvantEDGE MEC solution. Referring to a video service, the paper shows a scheme developed during the experimental study to allow the pod migration when K8s is used with AdvantEDGE. Using the emulation of user mobility given by AdvantEDGE platform, the described experimental tests allow to show the performance of the MEC application migration.

Keywords: Multi-Access Edge Computing (MEC) · AdvantEDGE · 5G-MEC Emulation · Kubernetes · Migration

1 Introduction

Due to the increasing demand for computational and Internet of Things (IoT) applications, the fifth generation (5G) of mobile networks will face unusual traffic volume. At end-users, computing-intensive applications become an inherent concern due to the end user's limited storage and computational abilities. Multi-access Edge Computing (MEC) is the emerging technology in the 5G network that can process a large amount of data within the Radio Access Network

This work was supported by the Norwegian Research Council through the 5G-MODaNeI project (no. 308909).

S. Yu et al. (Eds.): TridentCom 2022, LNICST 489, pp. 104–118, 2023.
https://doi.org/10.1007/978-3-031-33458-0_8

(RAN) [1]. European Telecommunications Standards Institute (ETSI) provides the MEC specifications to meet the requirements of the applications where real-time processing is needed. The core idea of MEC is to deploy the cloud computing capabilities within the RAN, close to the end-user [2]. ETSI MEC Ecosystem [3] refers to the MEC solutions that support the experimentation and deployment of the practical scenarios that include a 5G-MEC framework. In those MEC solutions, a recent study [4–8] shows AdvantEDGE as a potential emulation platform tool to perform the different challenges in the MEC framework.

AdvantEdge [9] is a Mobile Edge Emulation Platform (MEEP) that runs on Kubernetes (K8s) [10] and Docker [11]. The emulation platform enables the analysis with edge computing technologies, applications, and services. AdvantEdge provides the ability to explore edge deployment models, and allows the user to modify the deployment scenarios considering elements such as network topology, network characteristics, application mobility, and UE movement. AdvantEdge provides the connection of real cloudlet and UE applications so that simulation can capture the impact of network design on application performance. AdvantEdge also allows the measurements collection in InfluxDB time series [12]. InfluxDB is a time series database built specifically for storing time series data.

Virtualization technologies support the deployment and management of the MEC applications and the MEC host (MEH). K8s is developed by Google [13] and is a superior technology for automating the management, scalability, and deployment of containers and nodes. Containers are prevalently used for running stateful applications. However, the standard K8s don't have an in-built mechanism to migrate the stateful containers from one node to another.

The main contribution of the paper is the description of an experimental testbed aimed at evaluating the performance of MEHs migration strategies using the 5G-MEC emulation scenario implemented by the AdvantEDGE platform. Then, referring to a video streaming application, the paper presents an experimental analysis of the application migration in the runtime mode. The different time-related parameters related to the migration process are presented and analysed.

The paper is organized as follows. Section 2 gives the ETSI MEC application migration use case and the standardized ETSI MEC API for the application mobility using the AdvantEDGE platform. Section 3 presents the related work, while Sect. 4 discusses the main advantages of AdvantEDGE and the emulated network scenario considered for the experimental tests. Section 5 shows the testbed setup, the integration of the application migration techniques, and the working strategies. Section 6 evaluates the average values and confidence interval (CI) over the observation period of selected time-related performance parameters. The future work and the conclusions are summarized in Sect. 7.

2 Background

ETSI has specified the management of the MEC by considering the system level, the host level and the network layer functionalities. MEC Orchestrator (MEO)

is the brain and has the overall view of the MEC system level management elements. The MEC system level consists of the MEHs, physical resources, applications and its services along with system topology. The MEO is the responsible entity for selecting the MEH during the application instantiation for the end user. In the ETSI MEC architecture, a MEH has a MEC Platform (MEP), which can establish a connectivity with the other MEPs by using the Mp3 reference point. Mp3 reference point is the platform-to-platform interface that exchanges the information related to the application mobility between MEHs. In a disturbed deployment of the MEC system, multiple instances of the MEC application can be present and maintain the connectivity over different MEHs. The entities in the MEC application mobility within the intra-MEC system scope are presented in [14].

For the MEC application mobility, there are two entities to focus on: the application availability in the target host and the user context transfer. In the first entity, the application is required to be available in the targeted MEH, where the targeted MEH does not have designated application to provide the service to the end user. The MEO decides the application instantiation on the targeted MEH and has the ability to download the application image. The MEO can initiate the application by using the Virtualization Infrastructure Manager (VIM). After the application availability at the targeted host, a communication link is established to transfer the user context as the end user application is connected to the MEC application, the end user is not expected to be aware of the application mobility and the deployment of the application along with its state. A MEC stateful application needs to deliver the service continuity by importing the user context from the source MEH to the targeted MEH.

AdvantEDGE provides a support to the ETSI MEC API of the application mobility and allows the integration with the network scenario [15]. The API provides the support for relocation of user context between MEHs but the application instance relocation is not supported. The use case allows the MEC application user context transfer by using the API. The end-user devices are tracked and subscribed to the mobility procedure where the mobility manager receives the mobility notification of the end-user movement and the MEC application mobility. For running the application mobility experiment, the automation support is provided by AdvantEDGE for the User Equipment (UE) movement and the Point of Access (PoA) mobility.

3 Related Work

In [16], the authors discuss mobility-related issues, mainly focusing on the best instance to migrate the MEC application and what content to migrate to improve the Quality of Experience (QoE). Different mobility factors are taken into consideration. The work in [17] shows the optimal way to migrate the MEC application and the complete migration strategies to reduce energy expenditure. In [18], the authors consider the prototype system approach at the network layer to manage a seamless connection between the edge server and the mobile devices.

Some of the works carried out the experimental tests using different MEC model, and migration strategies, one of them presents K8s as the MEO [19]. The work proposes reactive service migration with the evolved packet core (EPC). Other experimental studies present the integration of Open Source MANO (OSM), an orchestrator, with Open Network Edge Services Software (OpenNESS) [20], a MEC platform, to migrate the MEC applications between MEHs [21]. The study includes two components, one to maintain the application's state with the client and the other to focus on management.

In [22], the authors discuss various container and migration strategies focusing on the fog, edge, and cloud; the work focus on the current approaches and the framework for container-based services migration. In [23], the authors describe different methods of the migration of pod in K8s and present the results of downtime with and without migration along with the data size transferred. For stateful container migration, a prototype approach using an extended version of kubelet and customized containers is available on GitHub [24]. The prototype approach provides an extension of the *kubectl* command that includes a command for the checkpoint and migration of the running pod in K8s. This work presents the running pod migration across single or multiple clusters and adds the function necessary to the pod migration. Furthermore, the prototype implementation includes pod migration operator at control plane that has custom resource and the controller.

4 Emulated Network Scenario

AdvantEDGE platform is a MEEP that provides emulated and experimental environment with edge enabling technologies [25]. The platform runs on Docker and K8s, and provides experimentation with MEC deployment models along with their applications and services. AdvantEDGE supports some of the APIs and the edge services standardized by the ETSI MEC such as ETSI MEC 013 Location [26], ETSI MEC 012 Radio Network Information [27], ETSI MEC 028 WLAN Information [28], ETSI MEC 011 Edge Platform Application Enablement [29] and ETSI MEC 021 Application Mobility [14]. In addition to that, AdvantEDGE allows the changes of the location of devices within the network using their own APIs. The platform allows network characteristics configuration such as latency, jitter, throughput and packet loss that can be applied to the scenario. During the scenario deployment, containers run in the K8s pod. In each deployed pod, AdvantEDGE includes an companion container called as sidecar. The role of the sidecar is to apply the network characteristics from the simulation model. To implement simulation model, *TC-engine* is the responsible micro-service, *tc* is called as Traffic Control. Whereas *tc-netem* technology is used for the network characteristics in each sidecar. AdvantEDGE supports different edge application and client deployment model. Furthermore, the platform allows mobility event, UE movement and mapping the geo-location of each elements. The UE movement can be monitored and visualized using the Geospatial Subsystem.

Figure 1 shows the emulated network scenario based on AdvantEDGE platform. The scenario consists of one UE (ue1) in zone1, whereas zone2 and zone3 include an emulated MEHs edge1 and edge2 respectively along with PoAs. There are three different network access technologies representing different zones. The ue1 is able to connect to the MEHs via PoAs within each zone depending on the ue1 movement. The blue boxes are the MEHs. The brown boxs represent the mec-app for MEC application deployed on edge1 and vlc1 for ue1. The green box is the physical UE. The antennas represent the PoAs. Initially the MEC application runs on the MEH edge1 connected to a point on the network called Zone2. The edge1 MEC application and services are running externally to the platform. AdvantEDGE provides support to integrate the external MEC application and services within the scenario using an IP address and the port number. The zone elements represent a subnet, which can be composed by a set of network elements offering traffic transport service. Since the MEC architecture can be applied to any network technology, it is necessary to assume that further network elements are interposed between the infrastructures of an Internet Service Provider (ISP) and the edges of the network. These network elements are enclosed in logical zones and are not simulated by AdvantEDGE. In the Fig. 1, Operator1 is the ISP, which in the considered network scenario, provides the IP connectivity through the three access technologies and the IP services supported by means of the MEC architecture.

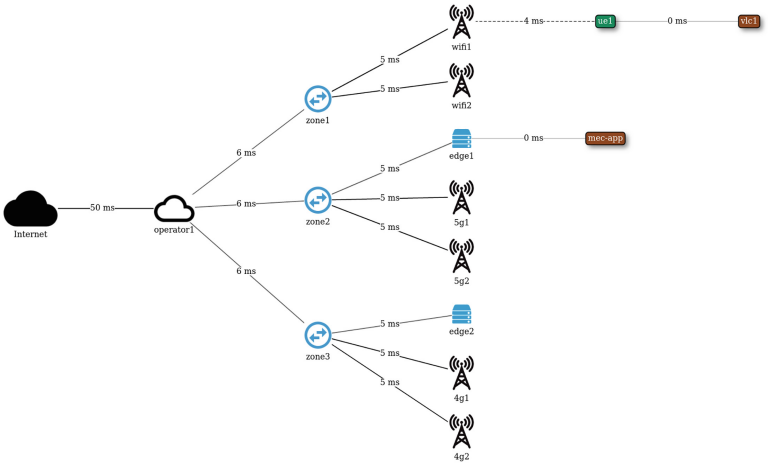

Fig. 1. Network scenario described by the AdvantEDGE GUI (Color figure online)

AdvantEDGE platform supports to allocate the physical locations of each elements presented in the scenario. The three different PoAs networking technologies are mapped in different geographical locations. Reference to the geographical scenario is in the Fig. 2. The scenario is an Arno River area in Pisa. The scenario considers three different access technologies: 4G, 5G and WiFi. The coverage radius of WiFi access technology is 200 m (in red), whereas is 500 m and 1000 m for 5G (in orange) and 4G (in blue) respectively. In the figure, the blue line denotes the ue1 path considered for the experimentation.

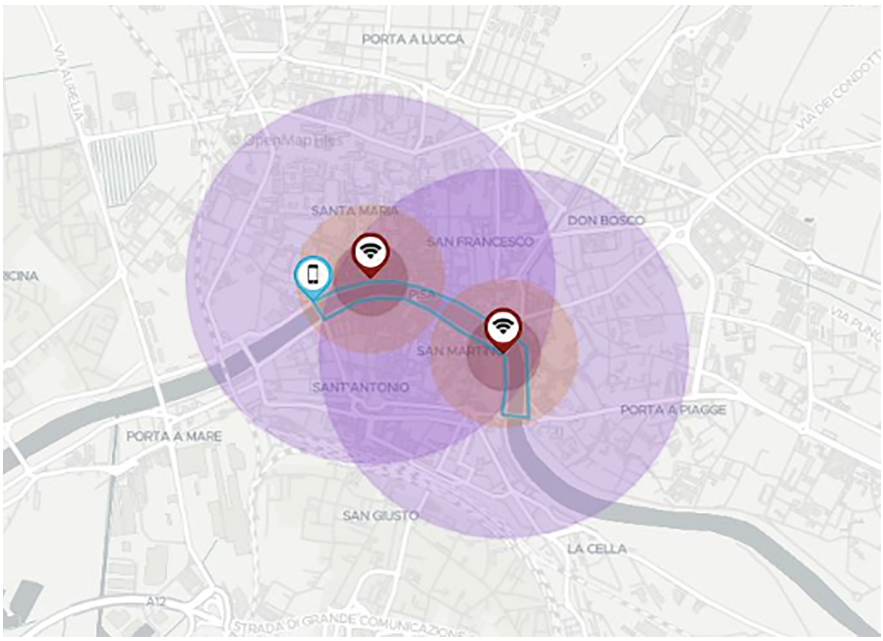

Fig. 2. Map of the scenario considered in the experimental analysis with AdvantEDGE platform. (Color figure online)

The value of each metric of the determined optimum path between the MEH and the UE, as a function of the chosen PoA, are summarized in Table 1.

Table 1. Parameters of the path for the experimental tests

MEH-UE	WiFi-1	4G-1	5G-1	WiFI-2	4G-2	5G-2
PLoss (%)	0	0.016	0	0	0.0079	0
Jitter (ms)	12	16	13	16	15	17
Latency (ms)	26	31	30	26	39	30

The GIS API (getGeoDataByName) was utilized during the tests to gather data on the geographical locations of the network's devices. These statistics make it possible to determine the client-to-PoA distance, which is critical to identify the set of PoAs that can provide connectivity to the client. AdvantEDGE's Sandbox API (sendEvent) enables runtime PoA handover, which permits switching the PoA to which the client is connected.

5 Experimentation

The experimentation of the MEC application migration using the AdvantEDGE platform is divided into three phases. In the first phase, the deployment and the working structure are explained. The second phase deals the backend of the MEC application migration technique and integration with the AdvantEDGE platform. The third phase presents the experiment's migration flow, from the configuration deployment to the completion of the migration during the UE movement.

5.1 Description of the Testbed

Figure 3 shows the testbed overview with logical connectivity of the involved elements. The testbed is composed of 3 physical machines with specifications of GIGABYTE (32/512) Intel i7 NUCs. The NUC1 AdvantEDGE platform is deployed and runs the emulated network scenario implemented with AdvantEDGE and the UE application. As described in the Sect. 4, the emulated network scenario is composed by a set of APs, with three areas and two MEHs, edge1 and edge2. NUC2 enforces edge1 MEH where initially the mec-app is deployed and later migrated to the NUC3 edge2 MEH depending upon the UE movement. The AdvantEDGE platform is installed on a single K8s node on Ubuntu 20.04.4 LTS Operating System (OS). The AdvantEDGE platform GUI is accessed using the IP address 152.94.64.68, through which the emulated network scenario is configured and deployed. In the scenario configuration, the external mec-app is mapped with the edge1 mec-app using the IP address and the port number, called the external node integration. AdvantEDGE provides support for experimenting with external nodes and applications.

The management and deployment of the edge1 and edge2 MEHs are done using the cluster of K8s nodes: edge1 functions as the master node and edge2 as worker node. The MEHs interact with the AdvantEDGE platform using the API request and response provided by the AdvantEDGE platform. AdvantEDGE supports some of the ETSI MEC APIs. In particular, the location API, standardized by the ETSI GS as MEC 013 [26], is used to track the information related to the UE physical location in the network during the experimentation. The mec-app is a video streaming application in a container deployed using K8s. The K8s run the application and allow access to the service using the IP address and port number. As the mec-app is deployed on the edge1 MEH, the video streaming is always accessed through the edge1 IP address and the specified

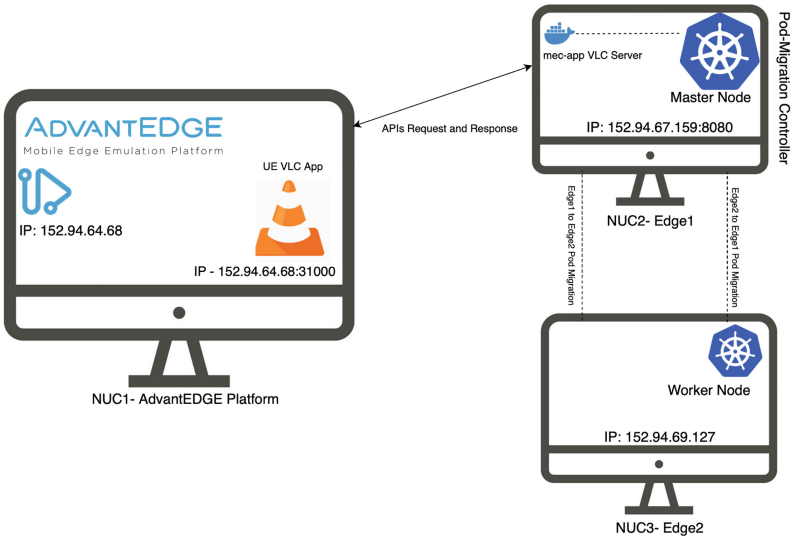

Fig. 3. The testbed

port number, as shown in the Fig. 3. The UE application is a VLC application [30] running on the NUC1. The UE application reaches the mec-app video streaming service through AdvantEDGE. The mec-app service maps within the AdvantEDGE platform, where AdvantEDGE creates a mec-app and UE app pod using the external node integration.

5.2 Migrating MEC Application

The MEHs edge1 (master node) and edge2 (worker node) form a single cluster of nodes using K8s as depicted in Fig. 4. In the single cluster, edge1 deploys the mec-app, a video streaming pod. As initially in the emulated network scenario, the UE is connected to edge1; during UE movement mec-app is migrated to edge2. The single cluster node using the extended K8s version prototypical implementation is available on GitHub [24]. The prototype implementation includes components such as the K8s, containerd-cri with the extensions of CRIU, which is needed for runtime pod migration and podmigration-operator. The K8s insures the node synchronization within the cluster of nodes. The extended version of the K8s provides *kubectl-migrate* and *kubectl-checkpoint* commands [24]. In addition, edge1 is configured as the NFS server, whereas the worker node is the NFS client. The NFS shared folder results into giving access to the edge1 checkpoint storage where the mec-app is running. The pod migration API server directs the pod migration from edge1 to edge2 or vice-versa. The podmigration controller includes Customized Resource Definition (CRD) and a custom controller to watch the pod migration within the cluster of the nodes. CRD is a mechanism that supports user-defined data types in K8s and permits to design the required

state while the controller can work towards the required state. The MEH edge1 runs the script to interact with the AdvantEDGE platform, where the UE information is exchanged using the APIs related to the AdvantEDGE platform and ETSI MEC specific.

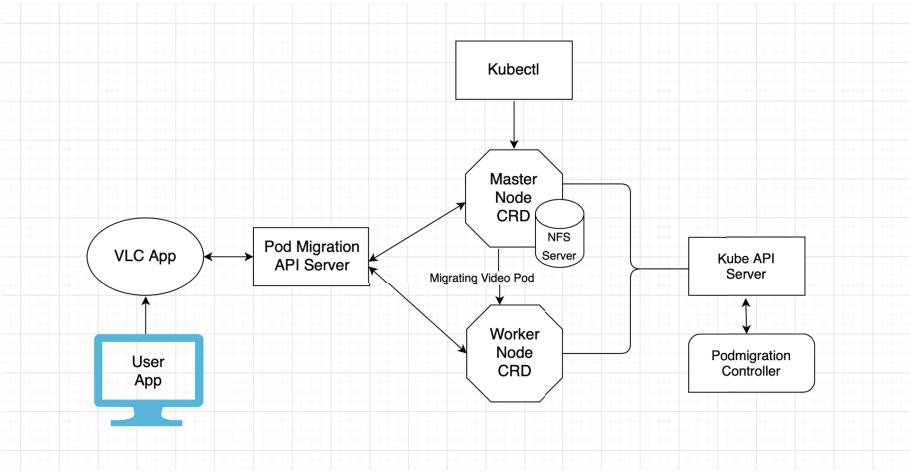

Fig. 4. Video pod migration

5.3 Migration Flow

Figure 5 presents the workflow of the MEC application migration between two MEHs using the AdvantEDGE platform. The emulated network scenario is created, configured, and deployed using the AdvantEDGE platform GUI. As soon as the scenario is deployed, AdvantEDGE creates a mec-app and UE app (referred to as vlc1) pod, which allows the UE app to reach the mec-app service via AdvantEDGE. The manager (script) registers the scenario information and the location of PoA and UE using the API. Initially, as configured in the scenario, vlc1 is closest and connected to the edge1 node. The vlc1 is connected, and the data is routed to the mec-app via edge1. The manager has pre-configured zone coverage for edge1 and edge2 depending upon the PoAs base station location. A PoA mobility event occurs during the UE movement, and the manager registers the UE location closer to the edge2. The manager orchestrates the scenario and guards the coordination with the podmigration-controller. The manager triggers and instructs the podmigration-controller for the mec-app migration from edge1 to edge2. The podmigration-controller starts the migration and checks if the source pod on edge1 is running or not. The podmigration-controller capture and

contain the container state in a pod. Once source pod running information is acquired, a checkpoint of the source pod is created at the edge2 along with the checkpoint path. The edge2 confirms the checkpoint info creation; then, the pod is restored at edge2. The edge2 informs podmigration-controller if the new pod is running or not. Once a new pod is running, vlc1 establishes the connection with the mec-app now running on edge2 via the manager. Later, the manager terminates the source pod with the help of the podmigration-controller. The manager affirms the app information with the UE application. In Fig. 5, the total migration time is noted from the start of the checkpoint to the completion of the migration process, whereas downtime is accounted for by resuming the pod on the destination MEH.

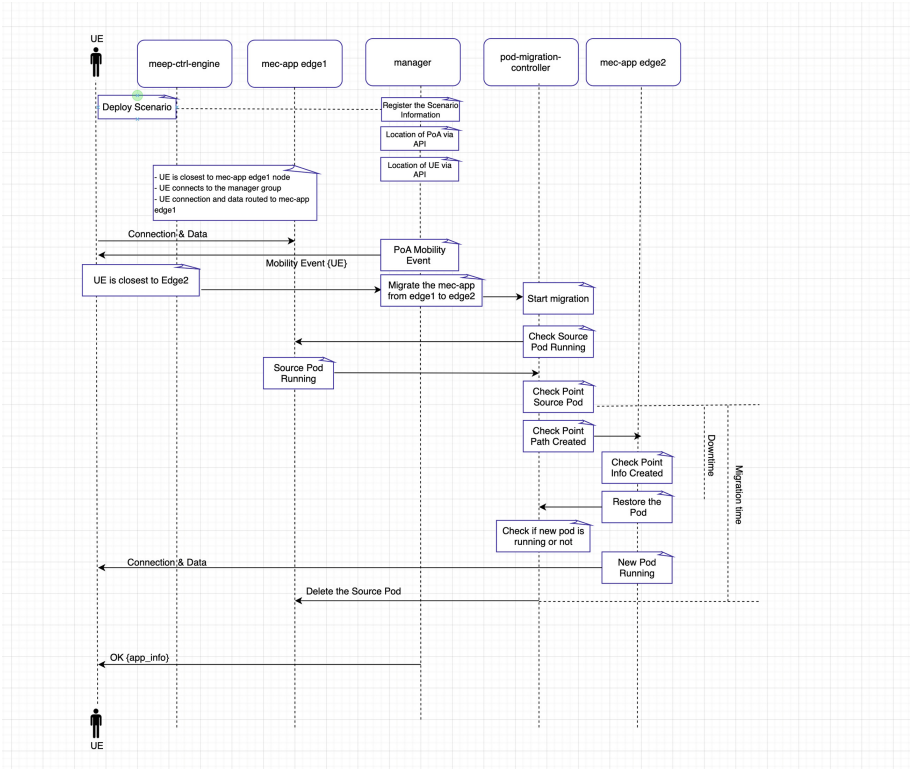

Fig. 5. MEC application migration flow

6 Experimental Results

The test aims to determine the pod migration capabilities of the extended K8s and the different time-related performance parameters. Figure 6 summarizes the different time-related parameters that can be measured during the experimental sessions. Furthermore, the observed values during only one migration are reported in the bottom line. The shown results refer to the video streaming application migration with the extended migration techniques. The test reports that the average MEC application migration time value is 3.762 s. The start of the delay is noted between the manager (script) to the pod migration operator. The start config delay mentioned in Fig. 6 relates to collecting information about running pods, such as whether the source pod is running or not and the current video state. The pod checkpoint delay is provided for establishing the checkpoint path with the destination host. The amount of time accumulated for the checkpoint pod's complete state is reported by the pod readiness delay. The destination host, who also produces the pod's container, gives the image download delay and container formation delay. The container start delay shows that the container is started and running for the end user. The pod migration operator deletes the source pod and records the total pod migration time in the logs. The extended K8s migration process indicates that this method allows a seamless migration process for video streaming applications. In comparison, the standard K8s is ill-advised for the seamless migration process of video streaming applications. Without the K8s plugin developed for migrating the application, the container migration via K8s adds more delay to this operation. At the same time, this problem could be avoided in the single cluster of nodes but is likely to affect the response time parameter in the multiple clusters within the Edge Datacenter.

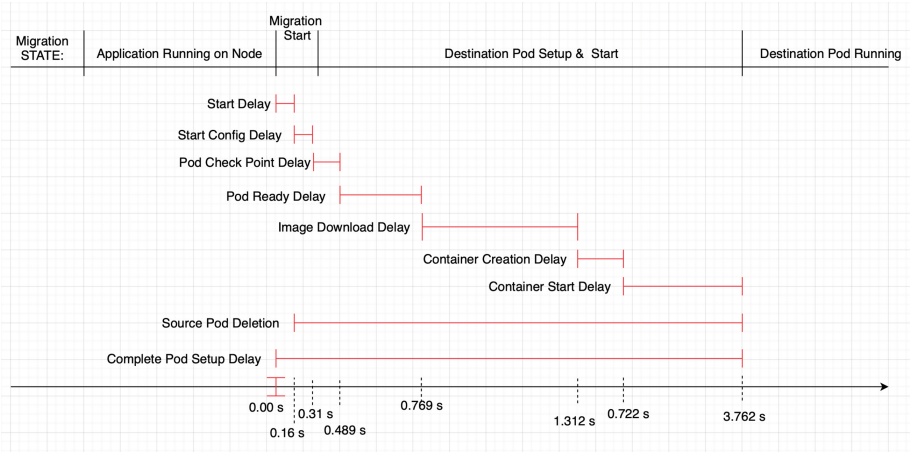

Fig. 6. Migration process in edge K8s

Table 2. Description of time measured in the experiments.

Δ (ms)	Total time taken for the application migration
Δ_1 (ms)	Time taken by the manager to start the migration
Δ_2 (ms)	Time accumulated for the Source Pod Checkpoint.
Δ_3 (ms)	Time for the Pod Ready Delay
Δ_4 (ms)	Time to restore the Pod at destination

Table 3. Time measurements (Averaging 100 independent migration executions)

	95% C.I.	Median	Min	Max	95-th percentile
Δ (ms)	3015.46 ± 134.533	2866.5	1459	6789	3903.7
Δ_1 (ms)	19.4857 ± 1.1836	18.0	10	35	29.0
Δ_2 (ms)	8.1714 ± 0.6100	8.0	4	16	13.0
Δ_3 (ms)	7.971 ± 0.6087	7.0	14	16	12.549
Δ_4 (ms)	2995.371 ± 112.016	2860.0	2230	4471	3688.65

A more deep analysis has been carried out observing a large set of migration events. In particular, these experimental tests are carried out between the two MEHs on the defined UE path as shown in Fig. 2. A total 100 number of migrations were taken place to observe the average migration time over the period. The extended version of the K8s and the NFS sharing helped to achieve a better and stable application migration latency. The MEC application migration time was noted from the pod migration operator logs with the help of the Kube API server. The time recorded in the logs depicts each stage of the pod migration from the creation of the checkpoint till the source pod's deletion. For the network layer, flannel is used. In K8s, the flannel supports the layer 3 networks between the multiple nodes across the single cluster, removing the port mapping complexities and providing the end user with a seamless migration experience.

Table 2 presents the set of time-related parameters that can be obtained from the K8s log, using the pod migration operator. The timeline dictate migration of the pod in the single cluster of nodes i.e. MEHs. Table 3 reports the statistical values of the Table 2 parameters during the experimental tests. The reported values refer to the statistics estimated observing 100 pod migration executions. Figure 7 shows the mean and the 95% CI of the total migration time taken by the pod migration operator, as a function of the considered number of pod migration executions. The figure points out the large CI when only 10 migrations are observed. In this case the values are ranged from 2.7 s to 4.5 s. On the contrary, after the observation of 100 migrations, the 95% C.I. has a size of only 134.53 ms around the mean value.

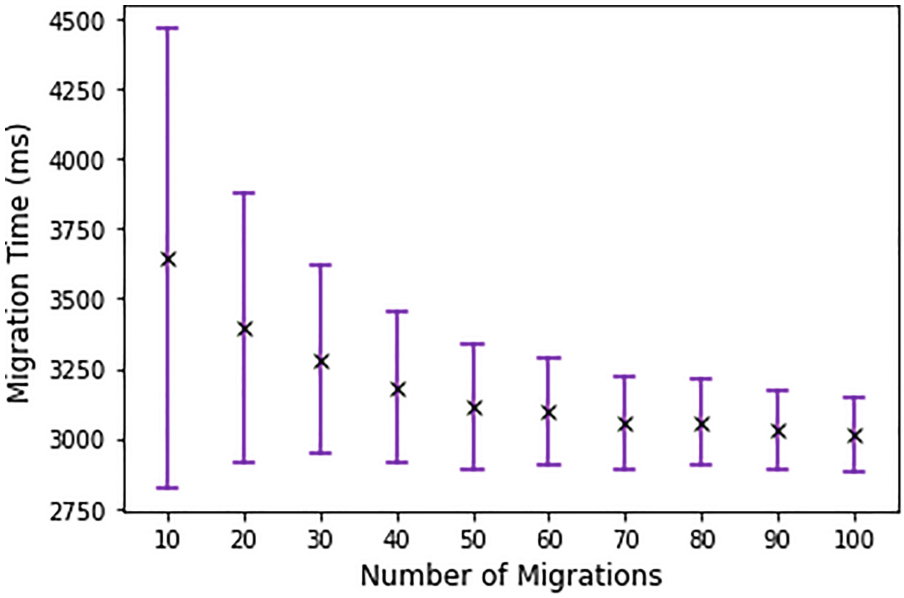

Fig. 7. Confidence interval of MEC application migration

7 Conclusion

In this paper, we presented and studied a scenario to migrate the MEC applications by using K8s and AdvantEDGE. The experimental study presents an analysis of the MEC application migration by using the extended K8s pod migration strategies. We evaluated the pod migration strategies for the MEC applications and the time related to the pod migrations between MEHs.

Acknowledgment. This work was partially supported by the Norwegian Research Council through the 5G- MODaNeI project (no. 308909) and the Italian Ministry of Education and Research (MIUR) in the framework of the FoReLab project (Departments of Excellence).

References

1. Pham, Q.V., et al.: A survey of multi-access edge computing in 5G and beyond: fundamentals, technology integration, and state-of-the-art. IEEE Access **10**(8), 116974–7017 (2020)
2. ETSI. ETSI GS MEC 003 V3.1.1: Multi-access Edge Computing (MEC); Framework and Reference Architecture (2022). https://www.etsi.org/deliver/etsi_gs/MEC/001_099/003/03.01.01_60/gs_MEC003v030101p.pdf
3. ETSI. MEC Ecosystem. https://mecwiki.etsi.org/index.php?title=MEC_Ecosystem. Accessed 4 Oct 2022

4. Blakley, J.R., Iyengar, R., Roy, M.: Simulating edge computing environments to optimize application experience. School of Computer Science Carnegie Mellon University, Technical report CMU-CS-20-135, November 2020
5. Gazda, R., Roy, M., Blakley, J., Sakr, A., Schuster, R.: Towards open and cross domain edge emulation - the AdvantEDGE platform. In: 2021 IEEE/ACM Symposium on Edge Computing (SEC), 14 December 2021, pp. 339–344. IEEE (2021)
6. Abdulmaksoud, M., Dehadrai, N., Castrillón, J., Sakr, A., Schuster, R.: Edge diagnostics platform: orchestration and diagnosis model for edge computing infrastructure. In: 2021 IEEE International Conference on Edge Computing (EDGE), 5 September 2021, pp. 51–59. IEEE (2011)
7. Burbano J, Sakr, A., Schuster, R.: Sliding-window approach for improving response time of mission-critical applications. In: 2020 IEEE 5th International Symposium on Smart and Wireless Systems within the Conferences on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS-SWS), 17 September 2020, pp. 1–7. IEEE (2020)
8. Sakr, A., Mohiyadeen, S., Vruksharaj, B., Schuster, R.: QoS-aware score-based edge resource allocation model. In: 2020 IEEE 5th International Symposium on Smart and Wireless Systems within the Conferences on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS-SWS), 17 September 2020, pp. 1–7. IEEE (2020)
9. Michel, R., Di Lallo, K., Robert, G.: AdvantEDGE: A Mobile Edge Emulation Platform (MEEP). GitHub. https://github.com/InterDigitalInc/AdvantEDGE. Accessed 4 Oct 2022
10. Kubernetes. https://kubernetes.io. Accessed 4 Oct 2022
11. Docker. https://www.docker.com. Accessed 4 Oct 2022
12. influxdata. InfluxDB https://www.influxdata.com. Accessed 4 Oct 2022
13. Google. Google Kubernetes Engine. https://cloud.google.com/kubernetes-engine. Accessed 4 Oct 2022
14. ETSI. ETSI GS MEC 021 V2.1.1: Multi-access Edge Computing (MEC); Application Mobility Service API (2020). https://www.etsi.org/deliver/etsi_gs/MEC/001_099/021/02.01.01_60/gs_MEC021v020101p.pdf
15. InterDigitalInc. AdvantEDGE. GitHub. https://interdigitalinc.github.io/AdvantEDGE/docs/overview/edge-services/ams/. Accessed 4 Oct 2022
16. Cruz, P., Achir, N., Viana, A.C.: On the edge of the deployment: a survey on multi-access edge computing. ACM Comput. Surv. (CSUR) **55**, 1–34 (2022)
17. Labriji, I., et al.: Mobility aware and dynamic migration of MEC services for the Internet of Vehicles. IEEE Trans. Netw. Serv. Manage. **18**(1), 570–84 (2021)
18. Kondo, T., Isawaki, K., Maeda, K.: Development and evaluation of the MEC platform supporting the edge instance mobility. In: 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), 23 July 2018, vol. 2, pp. 193–198. IEEE (2018)
19. Okwuibe, J., Haavisto, J., Harjula, E., Ahmad, I., Ylianttila, M.: Orchestrating service migration for low power MEC-enabled IoT devices. arXiv preprint, 30 May 2019. arXiv:1905.12959
20. Intel. Intel®Smart Edge Open. https://www.openness.org. Accessed 4 Oct 2022
21. Fondo-Ferreiro, P., et al.: Seamless multi-access edge computing application handover experiments. In: 2021 IEEE 22nd International Conference on High Performance Switching and Routing (HPSR), 7 June 2021, pp. 1–6. IEEE (2021)
22. Kaur, K., Guillemin, F., Sailhan, F.: Container placement and migration strategies for Cloud. A survey, fog and edge data centers (2022)

23. Schrettenbrunner, J.: Migrating Pods in Kubernetes (2020)
24. SSU-DCN. podmigration-operator. https://github.com/SSU-DCN/podmigration-operator/blob/main/init-cluster-containerd-CRIU.md. Accessed 4 Oct 2022
25. InterDigitalInc. AdvantEDGE. GitHub. https://github.com/InterDigitalInc/AdvantEDGE/tree/1e63a66e8820f0882c998f1cbc6d200bcd14f412. Accessed 4 Oct 2022
26. ETSI. ETSI GS MEC 013 V2.2.1: Multi-access Edge Computing (MEC); Location API (2022). https://www.etsi.org/deliver/etsi_gs/mec/001_099/013/02.01.01_60/gs_mec013v020101p.pdf
27. ETSI. ETSI GS MEC 012 V2.1.1: Multi-access Edge Computing (MEC); Radio Network Information API (2019). https://www.etsi.org/deliver/etsi_gs/MEC/001_099/012/02.01.01_60/gs_mec012v020101p.pdf
28. ETSI. ETSI GS MEC 028 V2.2.1: Multi-access Edge Computing (MEC); WLAN Access Information API. https://www.etsi.org/deliver/etsi_gs/MEC/001_099/028/02.02.01_60/gs_MEC028v020201p.pdf
29. ETSI. ETSI GS MEC 011 V2.2.1: Multi-access Edge Computing (MEC); Edge Platform Application Enablement. https://www.etsi.org/deliver/etsi_gs/MEC/001_099/011/02.02.01_60/gs_MEC011v020201p.pdf
30. Video LAN organization. Video LAN. https://www.videolan.org. Accessed 4 Oct 2022