Container-based Service Relocation for Beyond 5G Networks

Yeonjoo Lim, Jong-Hyouk Lee*

Sejong University, Republic of Korea
{yeonjoo, jonghyouk}@pel.sejong.ac.kr

Abstract

With the advent of 5G networks, various research on Multi-access Edge Computing (MEC) to provide high-reliability and ultra-low latency services are being actively conducted. MEC is an intelligent service distributed cloud technology that provides a high level of personal services by deploying cloud servers to edge networks physically closed to users. However, there is a technical issue to be solved, e.g., the service being used by a user does not exist in the new edge network, and there may even be situations in which the service cannot be provided in the new edge network. To address this, the service application must be relocated according to the location of the user's movement. Various research works are underway to solve this service relocation issue, e.g., cold/live migration studies have been carried in legacy cloud environments. In this paper, we propose a container migration technique that guarantees a smooth service application relocation for mobile users. We design scenarios for adaptive handoff and describe the detailed operation process. In addition, we present our MEC testbed, which has been used to experiment our container migration technique.

Keywords: Beyond 5G, Multi-access Edge Computing, Application Relocation, Service migration

1 Introduction

Mobile access technology is undergoing a revolutionary change every decade. Additionally, each generation of mobile access technology has delivered significant performance gains. Data usage has skyrocketed especially over the past decade due to the demand for high quality of video content. Users' viewing time is also increasing. Demand for content will continue to grow at a tremendous rate that surpasses forecasts. Since 2015, the 5th generation mobile communication standards established by the Radiocommunications Bureau of the International Telecommunication Union have been announced. Detailed technical discussions for IMT-2020 (5G) began in earnest in early 2016 [1]. Various studies are being conducted to satisfy 5G performance requirements. Among them, MEC considers service application deployments closed to the edge to provide ultra-low latency/high-performance services for users.

Application redeployment is being considered as a technique to improve users' quality of service. For instance, application redeployment can be integrated with a mobile user's handoff process to allow services to be relocated or deployed for a new edge network of the mobile user. Various schemes for application redeployment are being studied in recent years, e.g., a study to utilize live migration and cold migration technologies within virtual machines that exist in legacy cloud technologies. Recent research focuses on a container environment. If a user equipment (UE), which used a service on an edge network, has mobility, the same service may not exist or may not be provided on the moved edge network. To solve this, it is necessary to be able to redeploy the service application according to the location the user has moved to. In this paper, we propose a new migration technique for application redeployment. The proposed migration technique is based on container, which is much lightweight in terms of operation cost. We also present our testbed environment, wherein Kubernetes is used to effectively redeploy applications. As an experimental example, we show a stateful video streaming service performed in our testbed.

The rest of this article follows. Section 2 provides related works. Chapter 3 introduces the proposed method: MEC based service relocation. Chapter 4 presents the experimental results to evaluate the performance of the proposal. Section 5 concludes this paper.

2 Related Works

2.1 Multi-access Edge Computing

MEC has been proposed for high-reliability, low-latency communication in 5G networks and has been announced as a core technology to meet 5G KPIs [2]. Since 2014, MEC standards have been in progress centered on European Telecommunications Standards Institute (ETSI). Technical meetings for MEC standards are still active and ETSI Group Specification and ETSI Group Report are continuously developed [3].

MEC has been proposed to further meet 5G conditions by expanding research on fog computing among distributed cloud technologies. ETSI ISG is also conducting various Proof of Concept (PoC) projects. For real interoperability, project results follow standard specifications. MEC's initial name was 'mobile edge computing' for the purpose of supporting LTE. The name has been changed to 'Multi-Access Edge Computing' to support various access networks together.

In MEC, computing is done at the edge of the network rather than in the cloud or core network. The edge network exists between the radio access network (RAN) and the core network. Users move within the RAN and are served by MEC components within the edge network.

*Corresponding Author: Jong-Hyouk Lee; E-mail: jonghyouk@pel.sejong.ac.kr
DOI: 10.53106/160792642022072304026

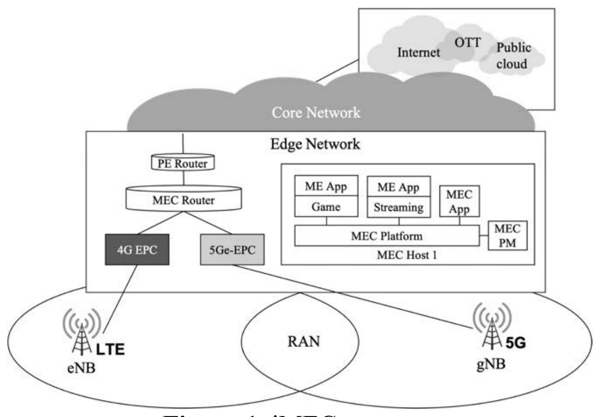


Figure 1. iMEC structure

Figure 1 above shows an MEC structure for a 4G and 5G coexisting environment. The characteristics of MEC are as follows: MEC has a distributed structure unlike the existing centralized cloud. It deploys an MEC host providing its own platform to run containerized applications. MEC also operates virtualization based MEC network configuration and management such as SDN and NFV.

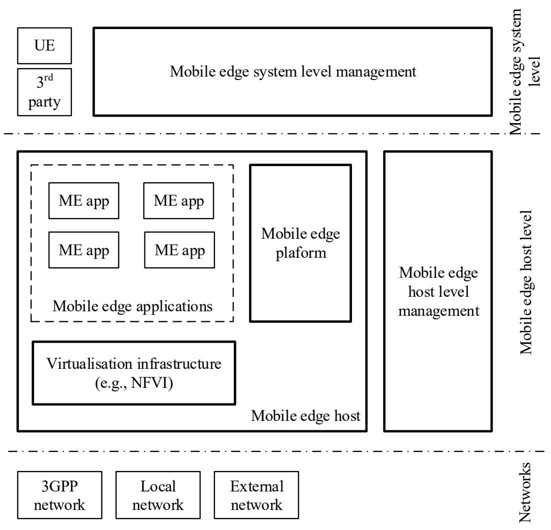


Figure 2. MEC framework [4]

As of August 2021, 48 standards have been published by ETSI MEC related groups. Among them, “MEC 002; Technical Requirements” and “MEC 003; Framework and Reference Architecture” documents are analyzed. It was published in 2016 to establish the initial concept of MEC. As shown in Figure 2 above, the MEC framework can be grouped at host level and MEC network level. MEC host includes MEC platform and one infrastructure that provides storage, network resources and storage devices for running MEC apps. MEC platform supports MEC apps in a specific virtualization infrastructure. A collection of essential features required to run and provide MEC services MEC apps are instantiated in the infrastructure of MEC hosts based on requests or environments implemented by MEC Platform Manager MEC Platform Manager is configured at MEC system level and MEC host level MEC system-level management includes the multi-access edge orchestrator as a core component with an overview of the entire multi-access edge system MEC host-level management manages the MEC platform administrator and specific mobile edge hosts and upper running application functions It consists of a Virtualization Infrastructure Manager (VIM).

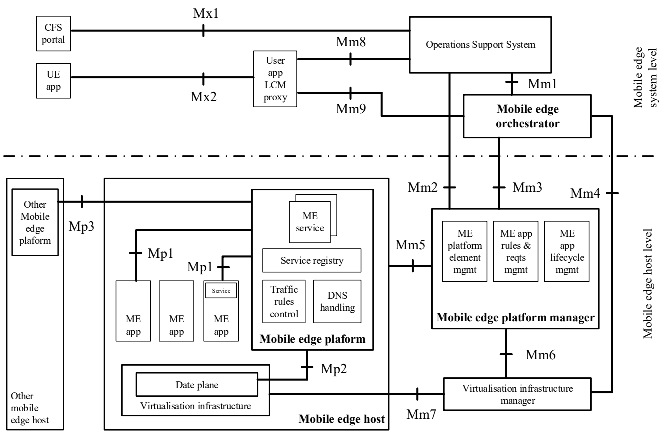


Figure 3. MEC RA [5]

The reference architecture shows the functional elements that make up the mobile edge system and the reference point between them. Figure 3 shows the mobile edge system reference architecture. Three groups of reference points are defined between system entities.

- Mp: Reference Points for Mobile Edge Platform Capabilities
- Mm: Management reference point
- Mx: Reference point connected to external entity

2.2 Application Relocation

As existing networks change to 5G MEC, one of the main features added is Application Relocation. With the 5G MEC environment, services for individuals (e.g., autonomous vehicles, telemedicine, and surgery) will increase. These applications need to keep users moving with them. To this end, the 5G MEC standard covers what is called Application Relocation [6]. Application relocation refers to a function in which the ME App must also be moved as the UE using the ME service moves. To do this, you must select the MEC server to be migrated when moving, and then perform a service migration for the application. It can be operated based on the service migration technique in the existing cloud environment. Service migration is a concept used in traditional cloud computing environments, formerly used to simply move to another infrastructure due to obsolescence of the infrastructure. Now, due to the expansion of cloud technology, service migration technology is being used in more diverse environments.

In an LTE network, handover is performed from an existing cell to another cell as a user moves. However, in 5G MEC, not only handover that a user moves, but also service migration according to mobility must be performed automatically. At this time, the time it takes from service termination to restart is called Service Down Time, and various methodologies are being studied to minimize the time. Research on this area is essential to satisfy the ultra-low latency, a key element of 5G.

2.3 Docker

Docker is an open-source container project launched in March 2013. The advent of Docker has made container research and implementation possible. Unlike traditional virtual machines, containers operate in isolated cgroups (control groups) and namespaces assigned by the host operating system. A cgroup is a feature of the Linux kernel that provides control (e.g., limits, isolation) over Linux system resources (e.g., memory, CPU, I/O, network) and allows the creation of resource groups for assigning ownership. Namespaces are also a feature of the Linux kernel that provides isolated spaces for filesystems, processes, networks, IPCs, hostnames, and users. In the beginning, Docker applied and operated Linux kernel's LXC (LinuX Containers) technology as it is, but from Docker 11, it is operated based on runC, which is implemented by itself based on OCI (Open Container Initiative) [7]. OCI is a project of the Linux Foundation to design open standards for operating system level virtualization (containers) and Linux containers [8]. Figure 4 below shows the difference between virtual machine and Docker architectures.

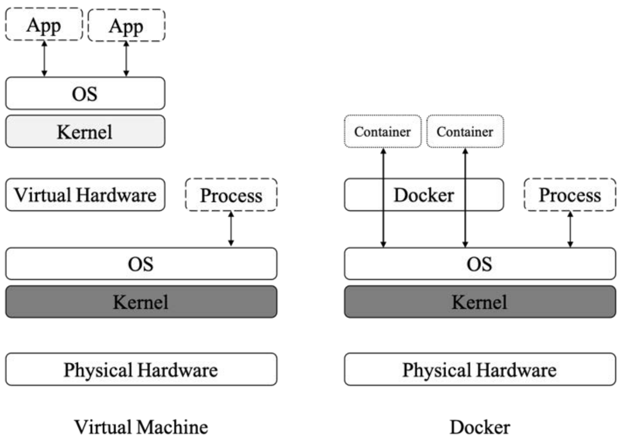


Figure 4. Virtual machine and docker architecture

Docker builds containers from images. Typically, images are uploaded to a remote repository and shared between users. Docker uploads and downloads to the remote repository via this name. And the image takes the form of several layers. When running containers, these layers are combined to act as a unified file system. It is called the union file system. Images have a base image layer and a container layer. The base image layer is the layer for running containers. Base images have read-only permissions, and multiple containers can be created with one base image. When the container is created, another layer is added to store the values changed by the user. This is called the container layer. A container layer is created above the base image. The container layer only stores created and modified files. When you convert this to an image, it is saved as a container layer stacked on top of the main image layer. Docker provides the ability to save the containers you've worked on as a new image. In other words, I created another base image layer by merging the container layer and the base image layer. The new image contains the running state of the old container. Docker provides various storage drivers and Docker Engine 20.10 works with overlay2 by default. Overlay2 is the preferred storage driver for all currently supported Linux distributions because it requires no additional configuration. Each storage driver has a different way of storing and mounting images when the container is created.

2.4 Kubernetes

Kubernetes is a system that can automate the management of containerized applications on multiple hosts and a framework that provides rich functionality to easily control the lifecycle of applications. In general, it provides a mechanism for deploying, maintaining, and scaling applications. This allows you to automate application management based on user-created policies. It aims to optimize container deployment management and works with Docker.

Kubernetes has clusters, which are collections of nodes for running containerized applications. Users can define different clusters depending on the service type. A cluster consists of one or more master nodes and worker nodes. A worker node is the host on which the actual application runs. The master node is the node that manages the worker nodes and the cluster. The master node handles the control plane, such as detecting all events in the cluster and determining actions. The following describes the application of the MEC environment. ME hosts are worker nodes that provide services to real users. And the ME manager where the master node performs monitoring and control nodes.

In Kubernetes, the smallest unit of an application is called a Pod. The smallest and simplest unit of a deployed object. Pods can be customized and configured by users in many different types. For example, you can create a single container or a pod that consists of multiple containers. Authorized nodes can monitor pod health in real time. Figure 5 shows the Kubernetes components [9].

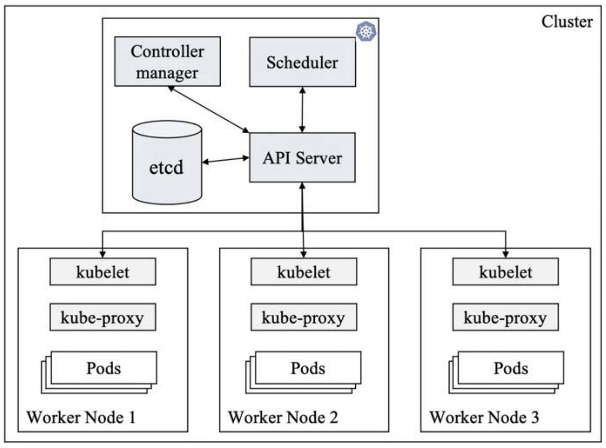


Figure 5. Kubernetes components

3 Proposed Service Relocation

3.1 Design

In the MEC architecture, a UE's cell handoff and the ME application's handoff occur. Here, we focus on describing our design for performing adaptive handoff in the MEC architecture. A handoff of the UE means that the UE moves to another cell. This occurs within the RAN while handoffs to ME applications occur within the edge network. This is called application relocation, which refers to moving application services between ME servers. When application relocation happening, there may be various ways to configure the service, but only the container case is considered in this paper.

There are two use cases where this application relocation occurs. The first case is Relocation After Preparation (Pro-active Migration): UE moves after synchronization of the base image. The second case is Relocation Before Preparation (Re-active Migration): UE moves without prior preparation. There is a difference between them, i.e., being active or being reactive for handoffs. In other words, application relocation can be classified depending on whether there is a preparatory procedure or not. In Pro-active Migration, the copy process is divided into two copies. The first copy syncs the base image and the second copies the container layer. In this paper, we call the base image as the pre-copy 1 and the container layer as the pre-copy 2. On the other hand, since there is no separate preparation procedure in Re-active Migration, there is only one image, and we call this as a post-copy. Note that in Re-active Migration, the service is resumed through one copy process after the terminal moves.

The concept defined in this paper has been well used in cellular network handoff scenarios. When a UE's handoff is detected, required preparations are proactively performed to minimize handoff latency. This is called a proactive handoff mechanism. Also, when a UE's handoff is not detected in advance, then a reactive handoff mechanism is performed to support the UE's handoff. In this paper, we further extend it to support smooth service relocation when the UE's handoff in the MEC architecture.

A diagrammatic representation of the service relocation algorithm is shown in Figure 6. As shown, depending on the handoff request message type, the service relocation algorithm decides Pro-active Migration or Re-active Migration. The following Sections 3.1.1 and 3.1.2 describe detailed procedures for each case.

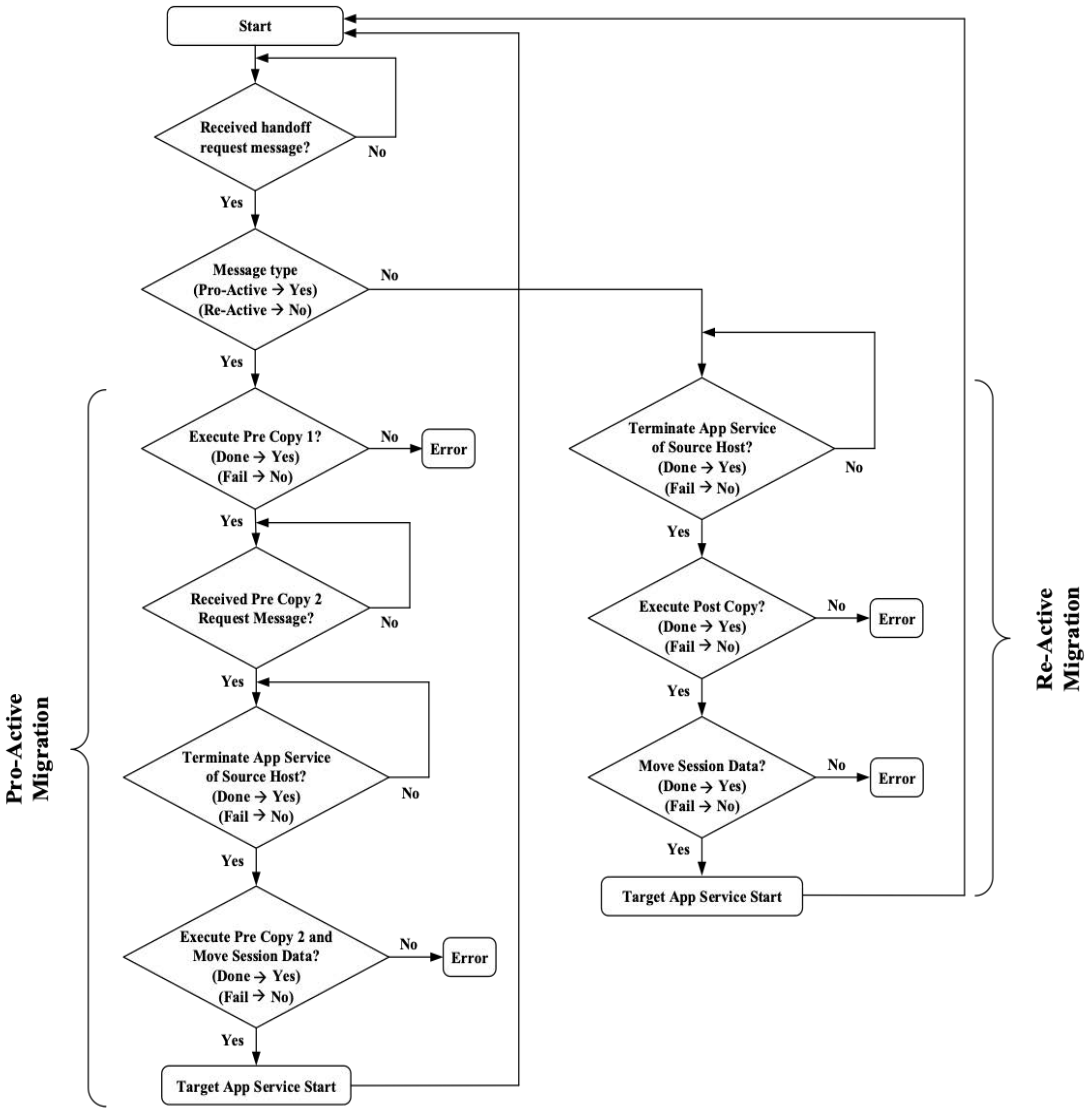

Start
Received handoff request message?
No
Yes
Message type (Pro-Active → Yes) (Re-Active → No)
No
Yes
Execute Pre Copy 1? (Done → Yes) (Fail → No)
No
Error
Yes
Received Pre Copy 2 Request Message?
No
Yes
Terminate App Service of Source Host? (Done → Yes) (Fail → No)
No
Yes
Execute Pre Copy 2 and Move Session Data? (Done → Yes) (Fail → No)
No
Error
Yes
Target App Service Start
Pro-Active Migration
Terminate App Service of Source Host? (Done → Yes) (Fail → No)
No
Yes
Execute Post Copy? (Done → Yes) (Fail → No)
No
Error
Yes
Move Session Data? (Done → Yes) (Fail → No)
No
Error
Yes
Target App Service Start
Re-Active Migration

Figure 6. Application relocation algorithm

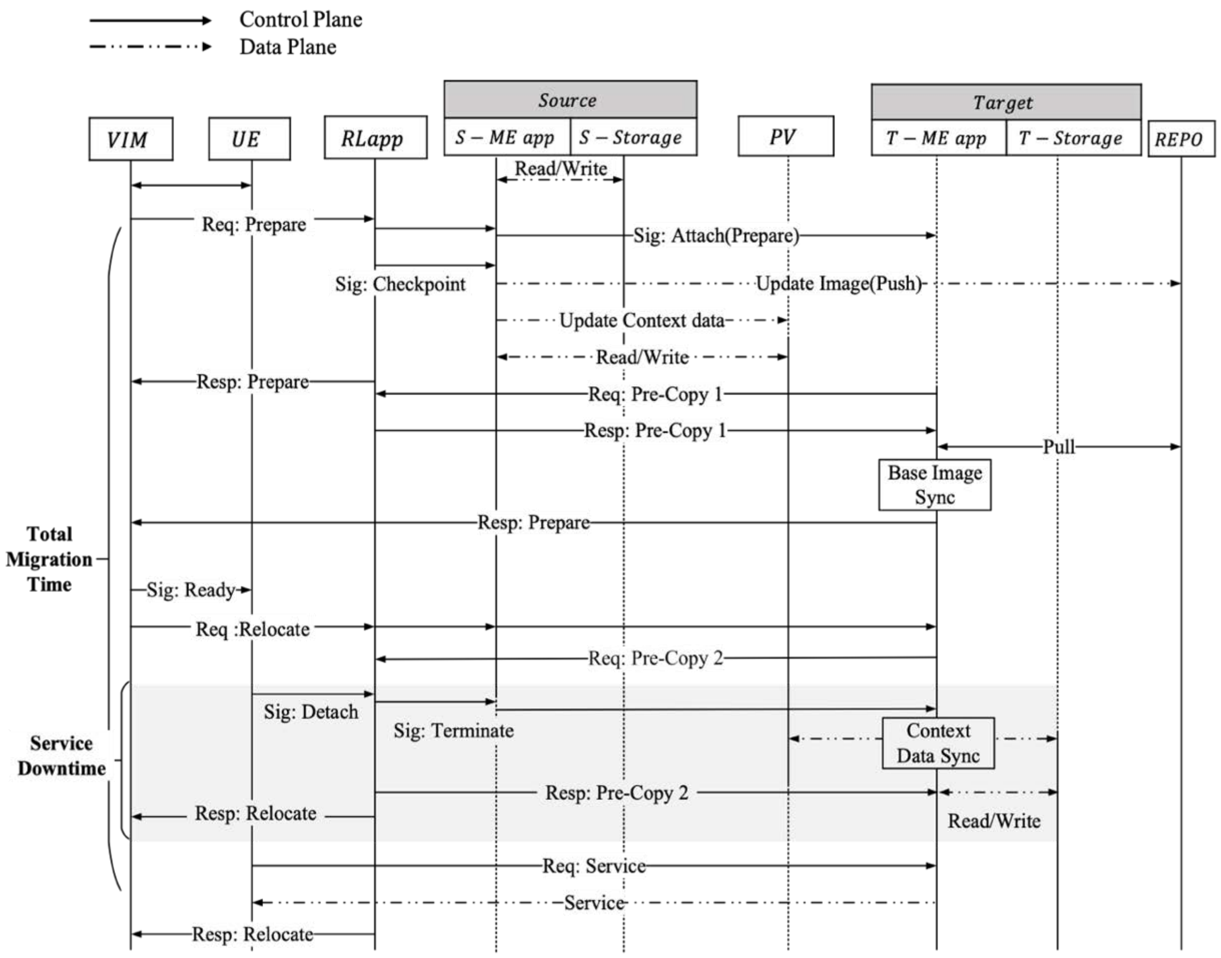


Figure 7. Pro-Active procedure

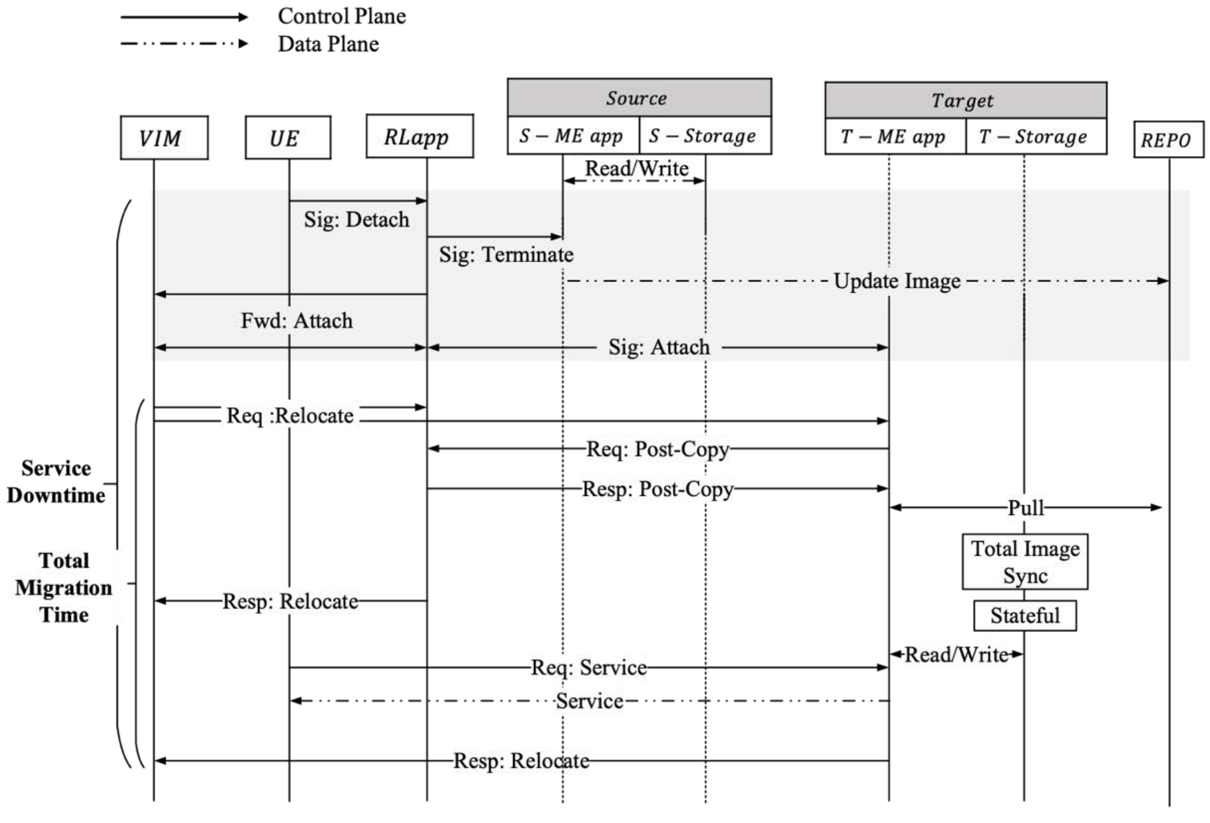


Figure 8. Re-Active procedure

3.1.1 Pro-Active Migration

This course starts with a pre-scenario once you decide whether you are ready or not. Therefore, the source host asks the target host for basic image synchronization procedures. This is called Pre-copy 1. The source application then stores the context data in a persistent volume (PV) until a handoff occurs at the UE. When this process is completed, the actual UE requests a connection to another cell and the ME host (target host). At this time, the service of the actual application is terminated, and service downtime begins. The target application runs a container based on the synchronized base image and synchronizes the data accumulated in the PV to the target storage. After completion of recovery, the target application notifies the end of Pre-copy 2 and restarts the service. The restart time is the endpoint of the last downtime measurement. Then, the UE receives the service from the moved host. Figure 7 above shows the detailed operation procedure.

3.1.2 Re-Active Migration

Reactive Process, on the other hand, has no preparation time. This scenario occurs in a situation after the actual UE moves to another cell. Similarly, the UE makes a connection request to the ME host (target host). At this point, the real application's service is shut down and service downtime begins. Commits the state of the source application just before shutdown and uploads it to the remote repository. And the target application downloads the uploaded application image. Start the service by running it as a container. The restart time is from when the source application is shut down to when the target service is restarted. Therefore, the post-scenario has more downtime than the pre-scenario. Figure 8 above shows the detailed operation procedure.

3.2 Implementation

We here describe the development of a service migration module that performs application relocation by merging the existing handover function and service migration function. That is, this paper proposes an adaptive service migration method according to the UE handover situation. The test bed was configured using Kubernetes. It is a form of deploying an ME app as a pod in Kubernetes, a container, and providing application services to the UE. The target service was selected as a video streaming service. Signing in initially allows users to watch videos and seamlessly use the same services as before, with the session they first logged in to, even after the user moves. The development was implemented only for the Pro-Active scenario. The notations and definitions of the components used are summarized in Table 1 below.

Table 1. Notation

Notations	Semantics in Kubernetes	Semantics in MEC	Definitions	Entity in Procedure
MEC-M	Master Node	ME host	Relocation Application	RLapp
ME Host 1	Worker Node	ME host	Server 1	Source
ME Host 2	Worker Node	ME host	Server 2	Target
MEApp	Pod	ME App	User Service	MEapp
RLapp	Pod	ME Application	Relocation Service	RLapp

The testbed consists of 3 ME hosts and 2 wireless RAN networks. Kubernetes builds 1 master node and 2 worker nodes that act as ME hosts. The band of the wireless RAN network was specified as 192.168.0.x for Cell 1 and 192.168.100.x for Cell 2. ME Hosts 1 and Cell 2 are in Cell 1 and Cell 2 respectively and are the closest servers. Each cell provides various ME App services to the UE. Figure 9 shows the schematic of the test bed. This is the case of service migration from ME App 2 on ME Host 1 to ME App 3 on ME Host 2. As the UE moves from Cell 1 to Cell 2, services are also migrated. A zone is the same concept as a cell. In Kubernetes terminology, being in a different zone means being in a different cluster. As previously designed in Section 3.1, development proceeded with both Pro-Active and Re-Active scenarios.

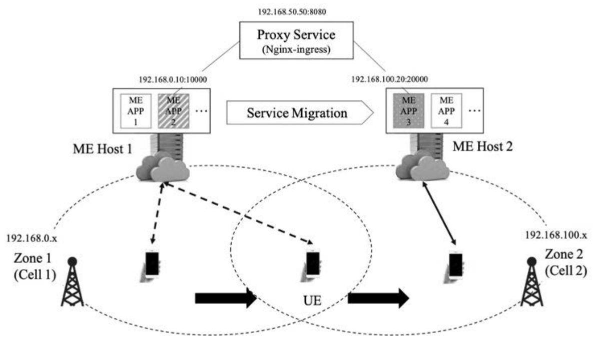
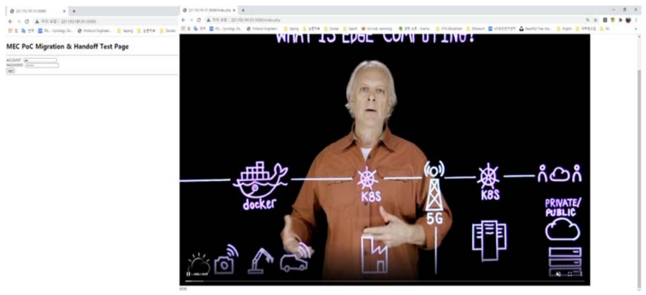


Figure 9. Network configure diagram

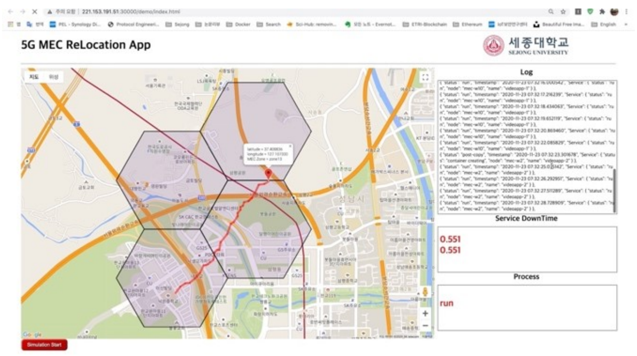
As shown in Figure 10, what the user is receiving is the web page of the video streaming server. The following describes the web screens on each side. On the left is the login interface. Initially, users log in to use the service. When the login is completed, the video is played from the beginning as shown on the right because this is the first login. Even if the service is restarted after the user moves, the logged-in session is maintained and the viewing time before the end of the service is saved. Seamlessly use the same services as before, after Application Relocation.


MEC PoC Migration & Handoff Test Page

ACCOUNT
PASSWORD

Figure 10. ME app (videoapp)

The following is web of RLapp. As shown in the Figure 11, Google Map API-based user movement path (latitude, longitude, node information) is displayed. Outputs RLapp (Relocation Application) server log, service downtime and service status/process. In the log, it displays the timestamp and the cell and zone to which the UE belongs and includes the actual GPS latitude and longitude.



```
n", "node": "mec-w10", "name": "videoapp-1" } },
{ "status": "run", "timestamp": "2020-11-23 07:32:17.216239", "Service": { "status": "ru
n", "node": "mec-w10", "name": "videoapp-1" } },
{ "status": "run", "timestamp": "2020-11-23 07:32:18.434063", "Service": { "status": "ru
n", "node": "mec-w10", "name": "videoapp-1" } },
{ "status": "run", "timestamp": "2020-11-23 07:32:19.652119", "Service": { "status": "ru
n", "node": "mec-w10", "name": "videoapp-1" } },
{ "status": "run", "timestamp": "2020-11-23 07:32:20.869460", "Service": { "status": "ru
n", "node": "mec-w10", "name": "videoapp-1" } },
{ "status": "run", "timestamp": "2020-11-23 07:32:22.085829", "Service": { "status": "ru
n", "node": "mec-w10", "name": "videoapp-1" } },
{ "status": "post-copy", "timestamp": "2020-11-23 07:32:23.301678", "Service": { "statu
s": "container creating", "node": "mec-w2", "name": "videoapp-2" } },
{ "status": "run", "timestamp": "2020-11-23 07:32:25.025142", "Service": { "status": "ru
n", "node": "mec-w2", "name": "videoapp-2" } },
{ "status": "run", "timestamp": "2020-11-23 07:32:26.292951", "Service": { "status": "ru
n", "node": "mec-w2", "name": "videoapp-2" } },
{ "status": "run", "timestamp": "2020-11-23 07:32:27.511289", "Service": { "status": "ru
n", "node": "mec-w2", "name": "videoapp-2" } },
{ "status": "run", "timestamp": "2020-11-23 07:32:28.728909", "Service": { "status": "ru
n", "node": "mec-w2", "name": "videoapp-2" } },
```

0.551
0.551

run

Figure 11. RLapp web page

4 Experimental Evaluation

This section describes the contents of the experiment to evaluate the performance of the implemented system. Assuming that the UE handed off the existing service before performing the handoff, the average time required to restart the service was checked by connecting the same service from the other host Worker node 2. Table 2 showing the hardware specifications used in the experiment is as follows.

Table 2. Experiment environment

No	Role	OS	CPU	Memory
1	Server (Master)	CentOS Linux release 7.7.1908 (64bit)	Intel® Xeon® Gold 6140 CPU @ 2.30GHz	232GB
2	Server (Repository)	CentOS Linux release 7.7.1908 (64bit)	Intel® Xeon® CPU E5-2630 v4 @ 2.20GHz	16GB
3	Server (Worker)	CentOS Linux release 7.7.1908 (64bit)	Intel(R) Xeon® Silver 4110 CPU @ 2.10GHz	32GB
4	Server (Worker)	CentOS Linux release 7.8.2003 (64bit)	Intel(R) Xeon® I5-4690 CPU @ 3.50GHz	16GB
5	Client (UE)	Microsoft Window10 Home (64bit)	Intel(R) Xeon® i7-8550U CPU @ 1.80GHz	16GB

The types and information of the software used in the experiment are shown in Table 3 below.

Table 3. Software information

No	Software name	Version	Related work
1	helm	3.1.2	Infrastructure
2	Kubernetes	1.19.3	Infrastructure
3	Docker Engine	19.03.8	Infrastructure
4	Nginx	1.16.1	videoapp, RLapp
5	Gitlab	11.9.6	Image Repository

The experiment was conducted a total of 5 times. The downtime calculation formula and repeated experimental process are as follows.

- Calculation formula: Downtime (s) = $B - A$
  - A: Service 1 end time
  - B: Service 2 restart time after completion of handoff

- Repeated experimental process
  1. Deploying all services: running videoapp and RLapp
  2. Watch video from videoapp
  3. Start the migration
  4. Check Downtime
  5. Check to maintain the watch time of the login session video
  6. Terminating all services: shutdown videoapp, RLapp

The results of the 5-repetition test are shown in Table 4 below. There is information and downtime of the services operated at the time of each experiment. The average downtime in this experiment was 0.514 seconds.

Table 4. Experiment result

No	Downtime (sec)	videoapp-1 Information (Pod, NAME, IP)	videoapp-2 Information (Pod, NAME,IP)
1	0.503	videoapp-1-c44cb848b-q4mf6 172.18.233.255	videoapp-2-7476c95669-62lgb 172.18.182.75
2	0.530	videoapp-1-c44cb848b-7gxq2 172.18.233.4	videoapp-2-7476c95669-z2tm4 172.18.182.76
3	0.490	videoapp-1-c44cb848b-g2pmh 172.18.233.5	videoapp-2-7476c95669-z7s6f 172.18.182.77
4	0.514	videoapp-1-c44cb848b-b4vv7 172.18.233.152	videoapp-2-7476c95669-995xt 172.18.182.78
5	0.533	videoapp-1-c44cb848b-rf4z5 172.18.233.154	videoapp-2-7476c95669-ng7qz 172.18.182.79

5 Conclusion

In this paper, we have presented an application relocation technique based on MEC. The proposal relocates the personalized service to a user according to the user's location. The proposed application relocation technique performs adaptively according to different scenarios and provides shorter latency through the process suitable for a selected scenario.

Acknowledgment

Jong-Hyouk Lee is a corresponding author. This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2022-0-00796, Research on Foundational Technologies for 6G Autonomous Security-by-Design to Guarantee Constant Quality of Security).

References

[1] ITU, *M.2083-0: IMT Vision – Framework and overall objectives of the future development of IMT for 2020 and beyond*, September, 2015

[2] ETSI White Paper, *MEC in 5G networks*, No. 28, June 2018.

[3] *ETSI Deliverables*, https://portal.etsi.org/Resources/Standards-Making-Process/ETSI-Deliverables, Accessed on July 2021.

[4] MEC, *GS MEC 003 - V1.1.1 - Mobile Edge Computing (MEC); Framework and Reference Architecture*, pp. 1-18, March, 2016.

[5] MEC, *GS MEC 003 - V2.1.1 - Mobile Edge Computing (MEC); Framework and Reference Architecture*, pp. 1-21, January, 2019.

[6] ETSI GS MEC 021 V2.1.1, *Multi-access Edge Computing (MEC); Application Mobility Service API*, January, 2020.

[7] Docker runC, https://blog.Docker.com/2015/06/runc/, Accessed on July 2021.

[8] OCI, https://opencontainers.org/, Accessed on July 2021.

[9] Kubernetes Components, https://kubernetes.io/docs/concepts/overview/components/, Accessed on July 2021.

Biographies

Yeonjoo Lim received a B.S. (Computer Engineering) degree from Sangmyung University, Cheonan, South Korea, a M.S. (Information Security) degree from Sejong University, Seoul, South Korea. She is currently a KT corporation researcher.

Jong-Hyouk Lee received the Ph.D. degree in computer engineering from Sungkyunkwan University, Suwon, South Korea. He was with INRIA, France, for IPV6 vehicular communication and security research. He was an Assistant Professor with TELECOM Bretagne, France. He was an Associate Professor with Sangmyung University, Cheonan, South Korea. In 2020, he moved to Sejong University, Seoul, South Korea. He is currently an Author of the Internet Standards: IETF RFC 8127, IETF RFC 8191, and IETF RFC 8691. His research interests include protocol engineering and performance analysis. He was a recipient of the Best Paper Award from the IEEE WiMob 2012, the 2015 Best Land Transportation Paper Award from the IEEE Vehicular Technology Society, the Haedong Young Scholar Award, in 2017, and the IEEE Systems Journal Best Paper Award from the IEEE Systems Council, in 2018. He was a Tutorial Speaker at the IEEE WCNC 2013, the IEEE VTC 2014 Spring, and the IEEE ICC 2016. He was introduced as the Young Researcher of the month by the National Research Foundation of Korea Webzine, in 2014.