applied sciences

MDPI

Article

Mobile Personal Multi-Access Edge Computing Architecture Composed of Individual User Devices

Juyong Lee [1], Jeong-Weon Kim [2] and Jihoon Lee [3,*]

1 Solution Research & Development Center, xService Business, Mobile Security Division, NSHC Inc., Geumcheon-gu, Seoul 08502, Korea; juyong.lee@nshc.net
2 Department of Professional Therapy, Gachon University, Seongnam-si 13120, Gyeonggi-do, Korea; zeezone@gachon.ac.kr
3 Department of Smart Information and Telecommunication Engineering, Sangmyung University, Cheonan-si 31066, Chungcheongnam-do, Korea
* Correspondence: vincent@smu.ac.kr; Tel.: +82-41-550-5441

Received: 11 June 2020; Accepted: 2 July 2020; Published: 5 July 2020



Abstract: The Multi-Access Edge Computing (MEC) paradigm provides a promising solution to solve the resource-insufficiency problem in user mobile devices by offloading computation-intensive and delay-sensitive computing services to nearby edge nodes. However, there is a lack of research on the efficient task offloading and mobility support when mobile users frequently move in the MEC environment. In this paper, we propose the mobile personal MEC architecture that utilizes a user's mobile device as an MEC server (MECS) so that mobile users can receive fast response and continuous service delivery. The results show that the proposed scheme reduces the average service delay and provides efficient task offloading compared to the existing MEC scheme. In addition, the proposed scheme outperforms the existing MEC scheme because the existing mobile user devices are used as MECS, enabling low-latency service and continuous service delivery, even as the mobile user requests and task sizes increase.

Keywords: multi-access edge computing; personal multi-access edge computing; multi-access edge computing mobility management; mobile edge computing; mobile edge computing mobility management

1. Introduction

The proliferation of high-performance user mobile devices and advances in network technology have led to the explosion of computational-intensive, delay-sensitive mobile application services such as real-time online games, virtual reality (VR) and augmented reality (AR) [1–6]. For example, AR services provide mobile users with an experience of interacting with the real world by adding computer-generated perceptual information to objects that exist in the real world. Since these services should quickly process data collected from cameras and sensors on user mobile devices, AR services require high computing power. Due to these changes in service and application requirements, mobile terminal technology at the edge of the network is also changing rapidly [7–9]. However, due to the limited battery capacity and computational processing power of the mobile user devices, mobile users cannot be provided with these services efficiently [10–12].

To meet these service and application requirements, the computing paradigm is widely used to deliver these services to mobile users using cloud computing such as Google Cloud Platform (GCP) and Amazon Elastic Compute Cloud (EC2) [13–15]. Cloud computing, which provides infrastructure as a service (IaaS), platform as a service (PaaS) and software as a service (SaaS) by clustering multiple data centers can deliver high-performance services that require large amounts of computation. In addition,

due to the various research and technology developments of the cloud computing, cloud computing can provide well-established deployment models and application platforms. However, the existing cloud computing architecture has long delays due to the propagation distance from the mobile user device to cloud data center. Also, the enormous amount of data exchanged between the mobile device and central cloud data center causes the data tsunami that saturates the backhaul network. These problems make it difficult to provide mobile users with the fifth-generation cellular network (5G) services that require high reliability and low latency such as ultra-reliable and low-latency communication (URLLC). Various kinds of network architectures and efficient schemes are emerging to solve these problems [16–19]. In particular, the small cloud computing containing the various computing functions close to the mobile user is the subject of active research [20–24].

The concept of fog computing, the service infrastructure that provides services similar to the cloud computing, is proposed by Cisco but does not act as a central server and acts like the cloud [25–27]. In other words, fog computing extends cloud computing to the edge to securely control and manage domain-specific hardware, storage, and network capabilities within the domain and secures powerful data-processing applications across the domain. Similarly, the concept of MEC proposed by European Telecommunications Standard Institute (ETSI) provides the powerful cloud computing functionalities in the radio access network (RAN) [28]. In other words, edge computing can provide services with relatively low latency, but the edge server has limited computing resources, which makes it difficult to perform tasks requiring high throughput. Although MEC-related studies are being actively carried out, there is a lack of research on edge cluster structures for efficient working offload [29–34]. However, when the edge server becomes congested due to frequent task offloading requests from mobile users, the advantages of edge computing that provide ultra-low latency response by performing computing tasks in close proximity to the mobile user disappear. Accordingly, there is a need for an efficient tasking offloading scheme in edge computing environments. To solve these problems, this paper aims to study how to support efficient task offloading by using user mobile device as edge computing host in the MEC environment.

The rest of the paper is organized as follows. Section 2 describes the proposed scheme's components and detailed operation process for using a user mobile device as an edge computing host. Section 3 presents the performance evaluation results of the proposed scheme. Section 4 concludes the result of the efficient and fast task offloading using mobile user devices as MECS.

2. Proposed Mobile Personal Multi-Access Edge Computing Architecture

The proposed scheme utilizes user mobile devices such as smart phones and laptops as MECS in addition to the existing MEC architecture. In addition to the existing architecture consisting of the MECS and cloud computing layers, it consists of the mobile MECS layer that utilizes mobile devices as MECS as shown in Figure 1. The proposed scheme utilizes user mobile devices as MECS, so that the user mobile devices enables the MECS and Cloud layer to distribute processing of computing loads. Therefore, this reduces the compute load at the MECS and cloud layers and the amount of data sent to higher layers to reduce bottlenecks. In addition, the scheme proposed in this manuscript utilizes the user mobile devices that exist closest to the mobile user as MECS, enabling rapid response and continuous service delivery, even when there are many users.

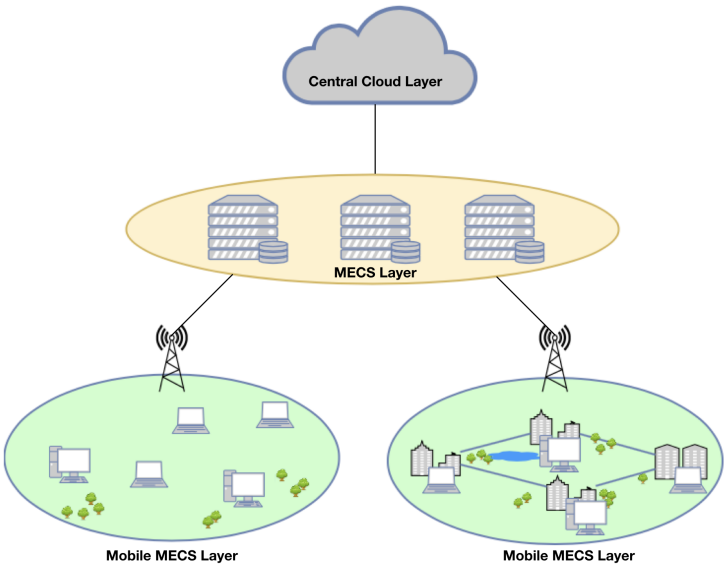


Figure 1. The design of the proposed mobile personal Multi-Access Edge Computing (MEC) architecture.

2.1. Components to Utilize Mobile Device as MECS

As shown in Figure 2, the components of the proposed scheme are the service requestor that requests the computing task, access MECS, which serves to generate task segmentation information by analyzing the requested task, and mobile devices, which act as a worker to perform tasks by receiving task segmentation information. In the proposed scheme, the mobile device used as the MECS is named mobile MECS (mMECS). The basic components and roles of the proposed scheme are shown in Table 1.

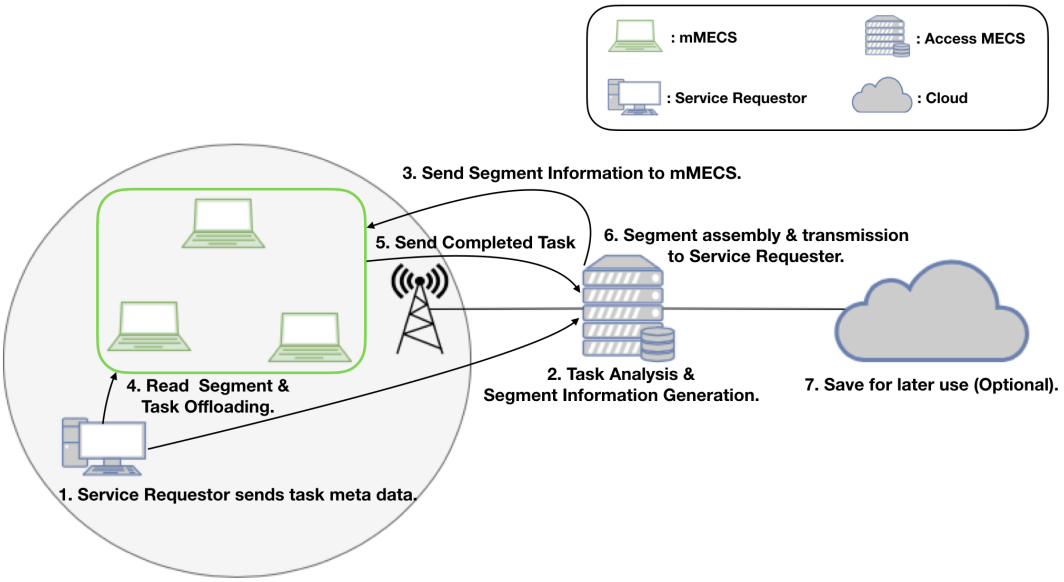


Figure 2. The components and operation steps of the proposed scheme.

Table 1. Description of the elements of the proposed scheme.

Component	Description
Access MECS	Creates the segment information of the task requested from the service requestor.
mMECS	Performs task offloading based on the segment information.
Service Requester	Serve task offloading requests to the access MECS.

The procedure for using MECS as the mobile device to provide fast response and continuous service is as follows. First, the service requester sends metadata of the task for requesting the task offloading to access MECS. The access MECS, which received the metadata, analyzes the requested task from the service requestor and generates segment information of the appropriate size. The generated segment information is transmitted to the mMECS, which has registered to be used as an MECS resource. Upon receiving of the segment information, mMECS receives the task segment from the service requestor and performs task offloading. The mMECS, which completed the task offloading of the task segment received from the service requestor, transmits the completed task to the access MECS. Access MECS, which has received the completed task segment from mMECS, reassembles the received task segment based on the segment information created by itself, and transmits the completed task to the service requester.

In other words, the proposed scheme uses the mobile device as an MECS in addition to the existing MEC architecture, so that it is possible to provide continuous and rapid service delivery even when the number of users increases. In addition, if the requested content is frequently used and highly shareable, the content is stored in the storage of the cloud layer and provided to the mobile user. Then the mobile user can receive the service from the cloud quickly even if the mobile user frequently requests the content. The detailed operational process of dispersion and reassembly of the task requested for computing offloading is described in Section 2.2. Also, the detailed operational procedures of the proposed scheme are described in Section 2.3.

2.2. Working Distribution and Merging Process

To maximize the distributed computing task processing speed, all the resources of mMECS, which plays the role of worker from the beginning to the end of the entire task execution process, should be utilized to ensure continuous operation. However, the proposed scheme utilizes mobile devices with diverse and limited computing power as MECS, so the existing computing method that distributes computing task considering all situations properly is not suitable. Since the proposed scheme utilizes the mobile node as MECS so there are a number of considerations for rearranging computing task when mMECS is moved. For this, the proposed scheme uses the method which sets the length of the segment to the smaller size so that mMECS, which is relatively poor in computing power, can be processed quickly. Figure 3 shows the task processing time of the distributed model based on task segment size.

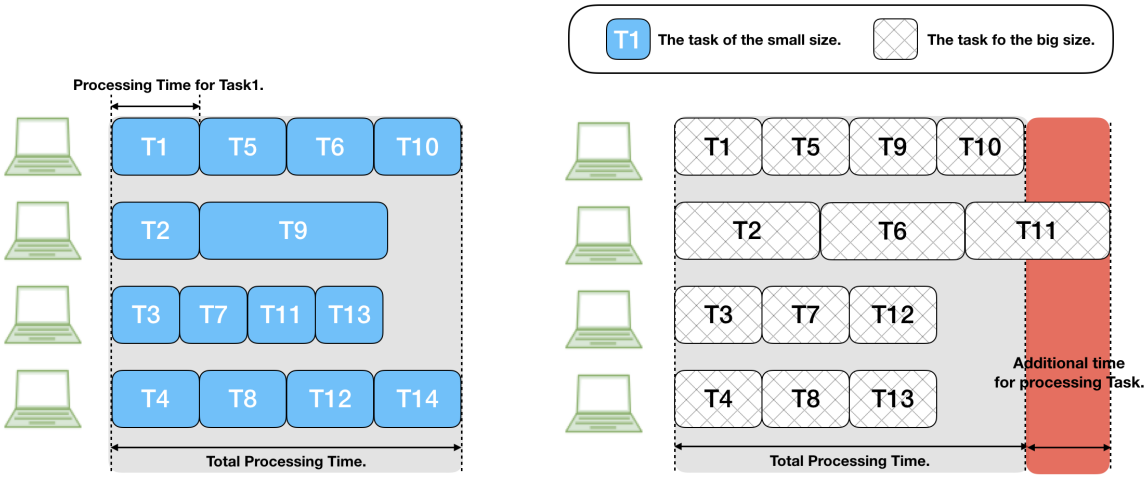


Figure 3. Task processing time in the task distribution model based on task segment size.

In the case of a task segment-based model, when a large segment size is set, bottlenecks and additional processing time may be caused since each of the mMECS with different computing capacities

has different task processing speeds. On the other hand, the distributed model used in the proposed scheme maximizes the task processing speed by allocating only a small number of task segments to mMECS at a time and assigning additional task segments right after the task processing is complete.

In other words, the proposed distributed model has a shorter segment length, so that the task distribution for the mMECS can be more balanced, and it is possible to use the worker's resources to the maximum.

2.3. Detailed Operation Process of the Proposed Scheme

The detailed operation process of the proposed scheme consists of three steps: mobile MECS registration step to utilize mobile node as MECS, service requester's request step for task offloading, and task offloading step of mMECS. A detailed description of the steps follows.

[Step01] Mobile MECS Registration: In order to provide the task offloading service to the service requester by using the mobile node existing in the access MECS as the MECS, the mobile node performs a procedure of registering its own information in the access MECS. To this end, the proposed scheme uses the mobile Node Registration (mNR) message to register the mobile node with the access MECS. The mNR message is a message format defined to register with access MECS that the mobile node will utilize its computing resources to play a role as the MECS. Figure 4 shows the format and example of the mNR message.

Mobile Node Registration Message

Mobile Node Name	Mobile Node Address	Type	Resource Field

Example> Mobile Node Registration Message

node01_laptop	223.33.175.233	**mNR**	Resource Field

Figure 4. The mNR message format and example.

The mobile node name and address fields of the mNR message each represent the name and address of the mobile node, and the type field is the field used by accessing MECS to identify a message. The resource field is a field that contains metadata that records computing power such as CPU performance and memory capacity of the mobile node. The mobile node that decide to provide the MECS service sends the mNR message to the access MEC. In the proposed scheme, the mobile node that provides MECS service by sending mNR message to access MECS is named mMECS. Upon receipt of the mNR message, the access MECS records the information for mMECS management in the mMECS Management Table (mMT) based on field information from the mNR message. Figure 5 shows the example of the mMT.

Mobile MECS Management Table

Name	Address	Resource	Status
node01_laptop	223.33.175.233	{cpu: […], memory: 4G, …}	False
node04_laptop	223.33.175.122	{cpu: […], memory: 8G, …}	True
node09_desktop	223.33.175.204	{cpu: […], memory: 1G, …}	True
node23_server	223.33.175.187	{cpu: […], memory: 2G, …}	False

Figure 5. An example of mMT generated by the access Multi-Access Edge Computing Server (MECS).

Since the proposed scheme utilizes mobile devices as MECS, it can provide low-latency service delivery than the traditional MEC architectures. However, since the mobile device is used as mMECS, it needs various considerations for continuous service delivery. Because the proposed scheme uses mobile devices as MECS, the access MECS needs to check whether or not mMECS has moved periodically. For this, the proposed scheme transmits segment information to mMECS and if it does not receive a response within a random time, it determines that mMECS has moved and retransmits segment information to another mMCES. In addition, in order to achieve optimal task offloading performance considering the different task processing speeds of mMECS, the proposed scheme utilizes the method for variable segment allocation. The variable segment allocation method works by allocating more segments to mMECS which can perform task offloading quickly.

For example, due to the slow computing task offloading processing speed of mMECS, when access MECS receives the processed task segment lately, access MECS adjusts the number of segments to provide fast task offloading to the service requester. In other words, the mobile device that has completed the registration in the access MECS by performing the above procedure can play a role as the MECS, so that it is possible to provide the faster service delivery to the requesting service than the existing MEC architecture.

[Step02] Service Requester Request for Task Offloading: In order for the service requester to request task offloading from mMECS, the proposed scheme initializes by sending metadata containing task offloading information to access MECS. For this purpose, the proposed scheme uses the task offloading request (TOR) message. The TOR message is used to request task offloading from access MECS by the service requester. Figure 6 shows the format and example of the TOR message.

Task Offloading Request Message

Service Requester Name	Service Requester Address	Type	Task META Field	Signature

Example> Task Offloading Request Message

user28_laptop	223.33.175.172	**TOR**	{type: video, task:encoding, info:	Signature

Figure 6. The task offloading request (TOR) message format and example.

The name and address field represent the mobile device name and address information of the service requester requesting the task offloading service. The type field is used by access MECS to identify the TOR message. The task meta field is a field that contains metadata information that is required for the task to be requested. Finally, the signature field is a field used for access control when the mMECS in the access MECS accesses the task. Based on the information in the meta field of the TOR message, the access MECS, which receives the TOR message, performs the task of determining the appropriate segment size using the variable segment allocation method as mentioned in Section 2.2. After determining the appropriate task segment size, the access MECS sends information on the service requestor and the segment information to the mMECSs suitable for performing task offloading by referring to the mMT.

[Step03] Perform Task Offloading of Mobile MECS: The access MECS, which determines the appropriate size of the task segment by referring to the mMT, uses the mobile service request (mSR) message to provide the mMECS with task offloading information to be performed. The mSR message consists of the service requestor's information and the task's information to be performed, which allows the mMECS to directly receive segmented data for task offloading from the service requester. Figure 7 shows the format and an example of the mSR message.

Mobile Service Request Message

Service Requester Address	Type	Task META Field	Signature

Example> Mobile Service Request Message

223.33.175.172	**mSR**	{type: video, task:encoding, info:…}[720p, …..]}	Signature

Figure 7. The mobile service request (mSR) message format and example.

The service request address field indicates the address of the service requestor to receive the segment required for the task offloading and the type field is the field used by mMEC to identify the mSR messages. The task meta field is a field that contains the task and segment information to be performed. The mMECS receiving the mSR message requests segment data assigned to it from the service requester, referring to the fields in the mSR message to perform task offloading. The service requester that receives the request of the segment data from mMECS sends the segment data and then that mMECS performs the task offloading. In order to merge the segment data that has completed task offloading and send it to the service requester, the proposed method needs to transmit the segment data completed by mMECS to the access MECS. To this end, the proposed scheme uses the mSR acknowledgement (mSRA) message. The mSRA message is used to send the segment that has completed task offloading to the access MECS. The mSRA message format and example are shown in Figure 8.

Mobile Service Request Acknowledgement Message

Mobile Service Request Message	Type	Data	Signature

Example> Mobile Service Request Acknowledgement Message

Mobile Service Request Message	**mSRA**	Data	Signature

Figure 8. The mSR acknowledgement (mSRA) message format and example.

The mobile service request message field indicates the mSR message transmitted by the access MECS to the mMECS, and the type field is a field used by the access MECS to identify the mSRA message. Finally, data filed is a field that contains segment data that has completed task offloading. The access MECS receiving the mSRA message reassembles the received segment data and transmits the assembled data to the service requester. In addition, to efficiently use mMECS computing resources, the access MECS performs periodic status checking of mMECS referring to its mMT and transmits segment information to other mMECS when the task offloading result is late or there is no response. Figure 9 shows the flow chart of the proposed Mobile Personal Multi-Access Edge Computing Architecture.

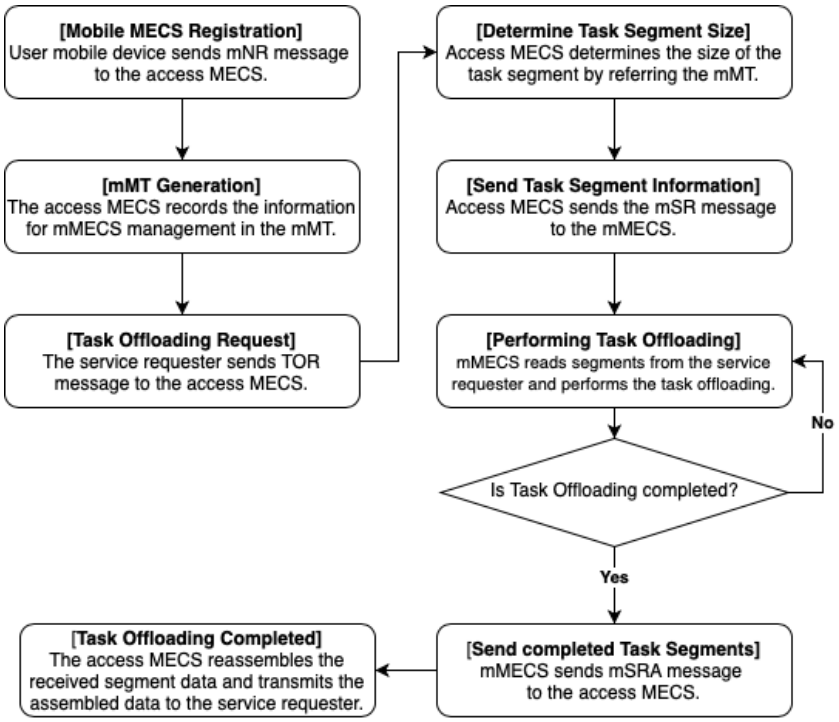


Figure 9. Flow Chart of the Mobile Personal Multi-Access Edge Computing Architecture.

3. Performance Evaluation

To evaluate the performance of the proposed scheme, we conducted experiments on various assumptions in which mobile users transmit computing tasks to access MECS and move to other access MECS. To do this, we implemented the network topology and MEC environments using CloudSim and Edge CloudSim respectively [35,36]. The mobile users moved according to a nomadic mobility model in which they move to other access MECS after a random amount of time has passed [37,38]. In our simulation model, there were three mobility types with different preference levels [39,40]. The mobile user was likely to spend more time in that access MECS in the case of low preference level.

In other words, the mobile user preference level of the location directly affected the dwelling time that the mobile user spent in the related access MECS. Million instructions per second (MIPS) was used to describe the ability to perform one million commands per second for the complexity of the computing operation. In addition, this paper handled the computation-intensive task and task offloading of mobile users using containerization technology, such as Docker, to handle tasks performed in simulation environments [41–43]. The percentages in the figures in this paper represent the usage of MEC resources. In Table 2, the important simulation parameters and their values are listed.

Table 2. Simulation Parameters.

Component	Description
User Mobility Model	Nomadic Mobility Model
Number of Mobile User	50–200
Number of Mobile Preference Level	3
Probability of Selecting a Mobile Preference Level	Equal (1/3)
Number of access MECS per Place	1
VM Processor Speed (MIPS) per access MECS	955
Processor Speed (MIPS) per mMECS	19
WLAN Bandwidth (Mbps)	300

3.1. Service Delivery Time for the Number of mMECS

In Figures 10 and 11, the service delivery time is shown with regard to the number of mMECS. For the performance evaluation, we set the task sizes to 3000 and 15000 MI and the results are shown in Figures 10 and 11, respectively.

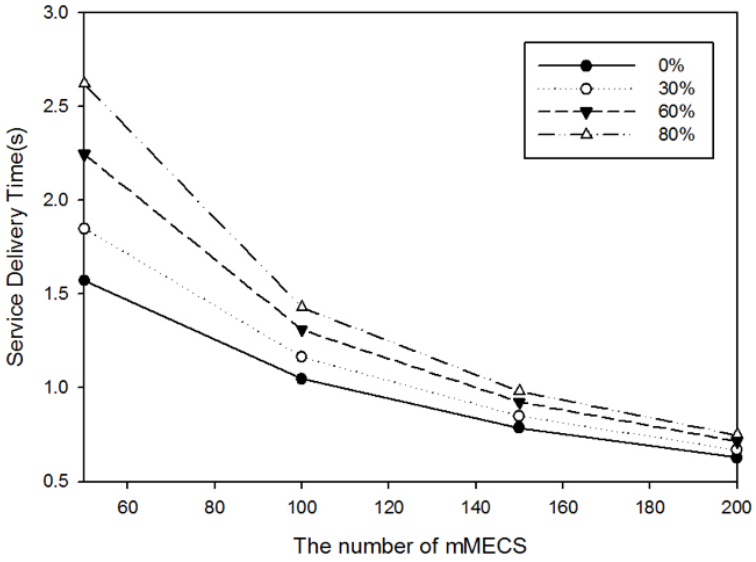


Figure 10. Service delivery time (Task sizes: 3000MI).

Figure 10 shows the service delivery time when the task size is small (3000MI). Since the proposed scheme uses a user mobile device as MECS, so it is possible to provide a fast service delivery even if the resources of access MECS are in use. Although access MECS can be handled without the help of mMECS due to the small size of computing task processing, it shows that it is possible to provide fast service delivery when the number of mMES is increased.

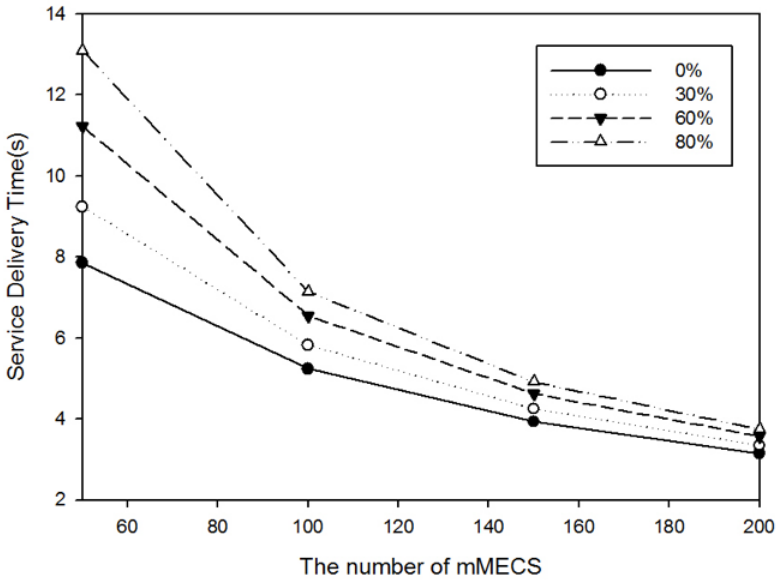


Figure 11. Service delivery time (Task sizes: 15000MI).

On the other hand, Figure 11 shows the service delivery time when task size is large (15000MI). In the traditional MEC architecture, access MECS, which is delegated a large size of computing task, cannot process other computing tasks until the completion of the delegated computing task. However,

the proposed scheme uses the user's mobile device as MECS, so it is possible to provide fast service delivery, even if the size of the task is increased.

3.2. Average Service Time Due to the Movement of the Mobile User

Figures 12 and 13 show the service delivery time when mobile users move frequently in relation to the number of mMECS. In order to measure the service delivery time according to the mobility of the mobile user, the mobility preference level is set to level 1 (low) and level 3 (high) and the results are as shown in Figures 12 and 13, respectively. Figure 12 shows the service delivery time when the mobility preference level is low. Even if the mobile user's movement is infrequent, if the mobile user moves, additional operations must be performed, so the service delivery time is longer than when the mobile user does not move. However, the proposed scheme utilizes the mobile user's device as the MECS, so if the number of mMECS increases, the service can be provided faster than the existing MEC architecture.

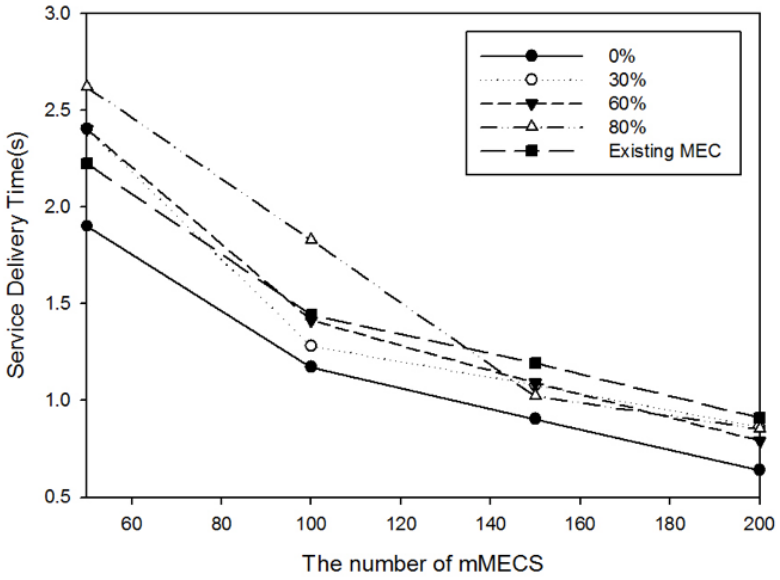


Figure 12. Service delivery time for the mobility preference: Level 1.

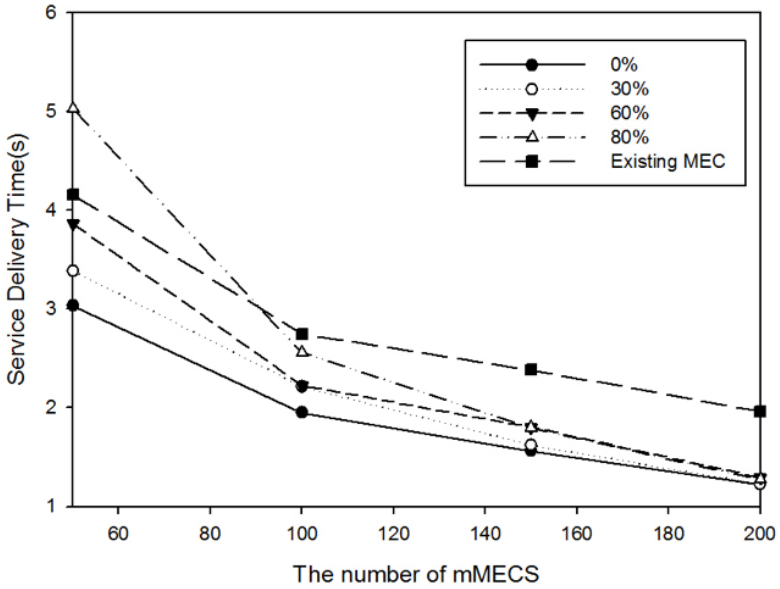


Figure 13. Service delivery time for the mobility preference: Level 3.

On the other hand, Figure 13 shows the service delivery time when the mobility preference level is high. In case of frequent movement of mobile users, it is impossible to provide fast service delivery because additional operation due to frequent movement and computation result should be received from previously located access MECS. However, the proposed scheme uses a mobile device existing in access MECS as mMCES, so it can provide faster service delivery than existing architecture in a new location. In addition, as the number of mMECS increases, the computing power increases, so it is possible to provide a fast service to mobile users.

4. Conclusions

This paper makes the following points. First, it shows that the existing MEC architecture brings computing capacity to the edge of the mobile network, which enables the mobile user to run applications that require ultra-low delay service to meet strict service requirements. However, when the mobile user requests and task size increases, there are problems that the existing MEC environment does not consider, such as continuous service delivery, fast response and efficient task offloading. To solve these problems, we propose a mobile personal MEC architecture that utilizes a user's mobile device as an MECS in this paper. The basic idea of the proposed architecture is designed to perform task offloading with the existing MECSs by using user mobile devices as MECS. In other words, the proposed scheme can efficiently support the task offloading by using the user's mobile devices as an MECS. In addition, the proposed scheme also allows mMECS to perform task offloading of mobile users, which enables continuous service delivery and rapid service delivery, even if mobile users move frequently. Finally, simulation results show that the existing mobile user devices are used as MECS, enabling low-latency service and continuous service delivery, even as the mobile user requests and task size increases. In future works, we will improve the proposed scheme by implementing predictive models based on machine learning (ML) to enable continuous service delivery and efficient task offloading, even if mobile users move frequently.

Author Contributions: J.L. (Juyong Lee) and J.L. (Jihoon Lee) conceived and designed the experiments; J.L. (Juyong Lee) and J.-W.K. performed the experiments; J.L. (Juyong Lee) and J.L. (Jihoon Lee) analyzed the data; J.L. (Juyong Lee) and J.-W.K. contributed reagents/materials/analysis tools; J.L. (Juyong Lee) and J.L. (Jihoon Lee) wrote the paper. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (No. 2020R1F1A1070215).

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

MEC	Multi-Access Edge Computing
MECS	Multi-Access Edge Computing Server
VR	Virtual Reality
AR	Augmented Reality
GCP	Google Cloud Platform
EC2	Amazon Elastic Compute Cloud
IaaS	Infrastructure as a Service
PaaS	Platform as a Service
SaaS	Software as a Service
5G	Fifth Generation
URLLC	Ultra-Reliable and Low-Latency Communication
ETSI	European Telecommunications Standard Institute
RAN	Radio Access Network
mMECS	mobile MECS
mNR	mobile Node Registration
mMT	mMECS Management Table

TOR	Task Offloading Request
mSR	mobile Service Request
mSRA	mSR Acknowledgement
MIPS	Million Instructions Per Second
ML	Machine Learning

References

1. Liu, Q.; Huang, S.; Opadere, J.; Han, T. An edge network orchestrator for mobile augmented reality. In Proceedings of the IEEE INFOCOM 2018-IEEE Conference on Computer Communications, Honolulu, HI, USA, 15–19 April 2018; pp. 756–764.
2. Sukhmani, S.; Sadeghi, M.; Erol-Kantarci, M.; El Saddik, A. Edge Caching and Computing in 5G for Mobile AR/VR and Tactile Internet. *IEEE Multimed.* **2018**, *26*, 21–30. [CrossRef]
3. Qiao, X.; Ren, P.; Dustdar, S.; Liu, L.; Ma, H.; Chen, J. Web AR: A Promising Future for Mobile Augmented Reality—State of the Art, Challenges, and Insights. *Proc. IEEE* **2019**, *107*, 651–666. [CrossRef]
4. Yang, X.; Chen, Z.; Li, K.; Sun, Y.; Liu, N.; Xie, W.; Zhao, Y. Communication-constrained mobile edge computing systems for wireless virtual reality: Scheduling and tradeoff. *IEEE Access* **2018**, *6*, 16665–16677. [CrossRef]
5. Elbamby, M.S.; Perfecto, C.; Bennis, M.; Doppler, K. Toward low-latency and ultra-reliable virtual reality. *IEEE Netw.* **2018**, *32*, 78–84. [CrossRef]
6. Lee, J.; Kim, D.; Lee, J. Mobile Edge Computing Based Immersive Virtual Reality Streaming Scheme. *Comput. Inform.* **2020**, *38*, 1131–1148. [CrossRef]
7. Wang, S.; Xu, J.; Zhang, N.; Liu, Y. A survey on service migration in mobile edge computing. *IEEE Access* **2018**, *6*, 23511–23528. [CrossRef]
8. Hao, Z.; Yi, S.; Chen, Z.; Li, Q. Nomad: An Efficient Consensus Approach for Latency-Sensitive Edge-Cloud Applications. In Proceedings of the IEEE INFOCOM 2019, Paris, France, 29 April–2 May 2019; pp. 2539–2547.
9. Palmarini, R.; Erkoyuncu, J.A.; Roy, R.; Torabmostaedi, H. A systematic review of augmented reality applications in maintenance. *Robot. Comput. Integr. Manuf.* **2018**, *49*, 215–228. [CrossRef]
10. Li, S.; Xu, L.D.; Zhao, S. 5G Internet of Things: A survey. *J. Ind. Inf. Integr.* **2018**, *10*, 1–9. [CrossRef]
11. Chen, M.H.; Dong, M.; Liang, B. Resource sharing of a computing access point for multi-user mobile cloud offloading with delay constraints. *IEEE Trans. Mob. Comput.* **2018**, *17*, 2868–2881. [CrossRef]
12. Zhou, B.; Dastjerdi, A.V.; Calheiros, R.N.; Buyya, R. An online algorithm for task offloading in heterogeneous mobile clouds *ACM Trans. Internet Technol. (TOIT)* **2018**, *18*, 23.
13. Challita, S.; Zalila, F.; Gourdin, C.; Merle, P. A Precise Model for Google Cloud Platform. In Proceedings of the 2018 IEEE International Conference on Cloud Engineering (IC2E), Orlando, FL, USA, 17–20 April 2018; Volume 10, pp. 177–183.
14. Ostermann, S.; Iosup, A.; Yigitbasi, N.; Prodan, R.; Fahringer, T.; Epema, D. A performance analysis of EC2 cloud computing services for scientific computing. In Proceedings of the International Conference on Cloud Computing, Munich, Germany, 19–21 October 2009; pp. 115–131.
15. Langmead, B.; Nellore, A. Cloud computing for genomic data analysis and collaboration. *Nat. Rev. Genet.* **2018**, *19*, 208. [CrossRef] [PubMed]
16. Jacobson, V.; Smetters, D.K.; Thornton, J.D.; Plass, M.F.; Briggs, N.H.; Braynard, R.L. Networking named content. In Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies, Rome, Italy, 1–4 December 2009; pp. 1–12.
17. Lee, J.; Lee, J. Efficient Mobile Content-Centric Networking Using Fast Duplicate Name Prefix Detection Mechanism. *Contemp. Eng. Sci.* **2014**, *7*, 1345–1353. [CrossRef]
18. Rosário, D.; Schimuneck, M.; Camargo, J.; Nobre, J.; Both, C.; Rochol, J.; Gerla, M.T. Service migration from cloud to multi-tier fog nodes for multimedia dissemination with QoE support. *Sensors* **2018**, *18*, 329. [CrossRef] [PubMed]
19. Lee, J.; Lee, J. Pre-allocated duplicate name prefix detection mechanism using naming-pool in mobile content-centric network. In Proceedings of the 2015 Seventh International Conference on Ubiquitous and Future Networks, Sapporo, Japan, 7–10 July 2015; pp. 115–117.

20. Li, H.; Shou, G.; Hu, Y.; Guo, Z. Mobile edge computing: Progress and challenges. In Proceedings of the 2016 4th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud), Oxford, UK, 29 March–1 April 2016 .
21. Bonomi, F.; Milito, R.; Zhu, J.; Addepalli, S. Fog computing and its role in the internet of things. In Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing, Helsinki, Finland, 17 August 2012; pp. 13–16.
22. Trakadas, P.; Nomikos, N.; Michailidis, E.T.; Zahariadis, T.; Facca, F.M.; Breitgand, D.; Rizou, S.; Masip, X.; Gkonis, P. Hybrid Clouds for Data-Intensive, 5G-Enabled IoT Applications: An Overview, Key Issues and Relevant Architecture. *Sensors* **2019**, *19*, 3591. [CrossRef]
23. Mach, P.; Becvar, Z. Mobile edge computing: A survey on architecture and computation offloading. *IEEE Internet Things J.* **2018**, *5*, 450–465. [CrossRef]
24. Lee, J.; Kim, D.; Lee, J. ZONE-Based Multi-Access Edge Computing Scheme for User Device Mobility Management. *Appl. Sci.* **2019**, *9*, 2308. [CrossRef]
25. Zuo, C.; Shao, J.; Wei, G.; Xie, M.; Ji, M. CCA-secure ABE with outsourced decryption for fog computing. *Future Gener. Comput. Syst.* **2018**, *78*, 730–738. [CrossRef]
26. Mahmud, R.; Kotagiri, R.; Buyya, R. Fog computing: A taxonomy, survey and future directions. In *Internet of Everything*; Springer: Singapore, 2018; pp. 103–130.
27. Bitam, S.; Zeadally, S.; Mellouk, A. Fog computing job scheduling optimization based on bees swarm. *Enterp. Inf. Syst.* **2018**, *12*, 373–397. [CrossRef]
28. Hu, Y.C.; Patel, M.; Sabella, D.; Sprecher, N.; Young, V. *Mobile Edge Computing—A Key Technology towards 5G*; ETSI White Paper; European Telecommunications Standards Institute: Valbonne, France, 2015; Volume 11, pp. 1–16.
29. Aloqaily, M.; Al Ridhawi, I.; Salameh, H.B.; Jararweh, Y. Data and service management in densely crowded environments: Challenges, opportunities, and recent developments. *IEEE Commun. Mag.* **2019**, *57*, 81–87. [CrossRef]
30. Lee, J.; Lee, J. Mobile Edge Computing based Charging Infrastructure considering Electric Vehicle Charging Efficiency. *J. Korea Acad. Ind. Coop. Soc.* **2017**, *18*, 669–674.
31. Balasubramanian, V.; Aloqaily, M.; Zaman, F.; Jararweh, Y. Exploring Computing at the Edge: A Multi-Interface System Architecture Enabled Mobile Device Cloud. In Proceedings of the IEEE 7th International Conference on Cloud Networking (CloudNet), Tokyo, Japan, 22–24 October 2018; pp. 1–4.
32. Wo, H.; Sun, Y.; Wolter, K. Energy-Efficient Decision Making for Mobile Cloud Offloading. *IEEE Trans. Cloud Comput.* **2020**, *8*, 570–584.
33. Chu, S.; Fang, Z; Song, S.; Zhang, Z.; Gao, C. Efficient Multi-Channel Computation Offloading for Mobile Edge Computing: A Game-Theoretic Approach. *IEEE Trans. Cloud Comput.* **2020**, *8*, 1. [CrossRef]
34. Joyee, S.; Ruj, S. Efficient Decentralized Attribute Based Access Control for Mobile Clouds. *IEEE Trans. Cloud Comput.* **2017**, *8*, 124–137.
35. Sonmez, C.; Ozgovde, A.; Ersoy, C. EdgeCloudSim: An environment for performance evaluation of edge computing systems. *Trans. Emerg. Telecommun. Technol.* **2018**, *29*, e3493. [CrossRef]
36. Calheiros, R.N.; Ranjan, R.; Beloglazov, A.; De Rose, C.A.; Buyya, R. CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms. *Softw. Pract. Exper.* **2011**, *41*, 23–50. [CrossRef]
37. Lee, J.; Lee, J. Hierarchical Mobile Edge Computing Architecture Based on Context Awareness. *Appl. Sci.* **2018**, *8*, 1160. [CrossRef]
38. Lee, J.; Lee, J. Preallocated duplicate name prefix detection mechanism using naming pool in CCN based mobile IoT networks. *Mob. Inf. Syst.* **2016**, *2016*. [CrossRef]
39. Pakusch, C.; Stevens, G.; Boden, A.; Bossauer, P. Unintended Effects of Autonomous Driving: A Study on Mobility Preferences in the Future. *Sustainability* **2018**, *10*, 2404. [CrossRef]
40. Wu, R.; Luo, G.; Yang, Q.; Shao, J. Learning individual moving preference and social interaction for location prediction. *IEEE Access* **2018**, *6*, 10675–10687. [CrossRef]
41. Bhimani, J.; Yang, Z.; Mi, N.; Yang, J.; Xu, Q.; Awasthi, M.; Balakrishnan, V. Docker container scheduler for I/O intensive applications running on NVMe SSDs. *IEEE Trans. Multi Scale Comput. Syst.* **2018**, *4*, 313–326. [CrossRef]

42. Merkel, D. Docker: Lightweight linux containers for consistent development and deployment. *Linux J.* **2014**, *239*, 2.
43. Felter, W.; Ferreira, A.; Rajamony, R.; Rubio, J. An updated performance comparison of virtual machines and linux containers. In Proceedings of the 2015 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), Philadelphia, PA, USA, 29–31 March 2015; pp. 171–172.



© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).