Latency Minimization in a Fuzzy-based Mobile Edge Orchestrator for IoT Applications

VanDung Nguyen, *Member, IEEE*, Tran Trong Khanh, Thant Zin Oo, Nguyen H. Tran, *Senior Member, IEEE*, Eui-Nam Huh, *Member, IEEE*, and Choong Seon Hong, *Senior Member, IEEE*

***Abstract*—Currently, matching the incoming Internet of Things applications to the current state of computing and networking resources of a mobile edge orchestrator (MEO) is critical for providing the high quality of service while temporally and spatially changing the incoming workload. However, MEO needs to scale its capacity concerning a large number of devices to avoid task failure and to reduce service time. To cope with this issue, we propose MEO with fuzzy-based logic that splits tasks from mobile devices and maps them onto the cloud and edge servers to reduce the latency of handling these tasks and task failures. A fuzzy-based MEO handles the multi-criteria decision-making process to decide where the offloaded task should run by considering multiple parameters in the same framework. Our approach selects the appropriate host for task execution and finds the optimal task-splitting strategy. Compared to the existing approaches, the service time using our proposal can achieve up to 7.6%, 22.6%, 38.9%, and 51.8% performance gains for augmented reality, healthcare, compute-intensive, and infotainment applications, respectively.**

***Index Terms*—Mobile edge orchestrator, edge computing, cloud computing, latency minimization, fuzzy-based approach**

I. Introduction

Recently, fifth-generation (5G) cellular technologies can further improve various new applications, such as video streaming analysis, augmented reality, the Internet of Things (IoT), and autonomous driving. However, these applications require real-time or semi–real-time computation while they create vast amounts of data and involve high user mobility, especially on the internet of vehicles [1]. Moreover, mobile users, which usually have limited processing capacity and battery life, cannot support these applications. Therefore, computation-intensive and delay-sensitive applications, a.k.a tasks, are suggested to rely on advanced computation offloading and the improved communication infrastructure to enhance the user's quality of service (QoS). Cloud computing can improve dynamic services and data-intensive analysis for mobile users over a wide area network (WAN). Nevertheless, the dramatically increasing amount of data generated by IoT devices is one of the major challenges in managing the traffic capacity of a WAN [2]. Additionally, because cloud computing works in a remote and centralized way, it cannot support context-aware computing for IoT applications [3]. Edge computing addresses these challenges by leveraging the distributed resources at the edge to provide timely and context-aware services. However, although an edge server at a base station (BS) has great potential to relieve the burden, its limited computational resources may not support all devices in its coverage under heavy traffic load services.

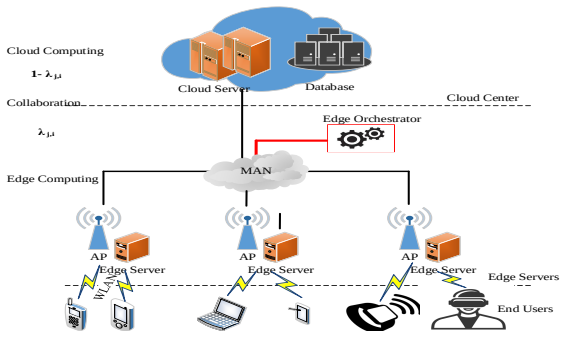

Fig. 1. MEO architecture and the role of an edge orchestrator.

To improve IoT systems, the mobile edge orchestrated architecture design integrates end devices, edge servers, and the cloud to form a hierarchical IoT architecture [4], as shown in Figure 1. The mobile edge orchestrator (MEO), also known as the application placement controller, plays a role in meeting application requirements by relying on controllers of other layers [5]. By using network information and matching it with the requirements received from applications, the orchestrator determines the target mobile edge (ME) host to process applications. Under IoT systems, it is necessary to design an edge computing system to handle a dynamic flow of requests that can be efficiently processed. Therefore, the workload orchestration problem must be studied in both computational and networking resources [4].

Several approaches have proposed orchestration in edge computing from different perspectives. In [6], the application deployment problem was studied to solve the problem under mobile edge computing. The MEO in [5] decides which application can be placed on which edge server. On the other hand, workload offloading by using the orchestration capabilities of software-defined networking (SDN) was proposed in [7], and workload offloading in the edge/fog computing infrastructure in [2] was studied. In [8], the authors proposed a heuristic offloading approach to enhance the capabilities of mobile devices. However, the studies mentioned above have the following drawbacks. First, an edge orchestrator cannot serve the entire system. Second, their work did not consider network congestion. Third, these approaches did not address

Manuscript received May 1, 2020; revised July 14, 2020 and August 13, 2020; accepted August 27, 2020. Date of publication xxx, 2020; date of current version xxx, 2020. This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the Grand Information Technology Research Center support program (IITP-2020-2015-0-00742) and service mobility support distributed cloud technology (MSIT-2017-0-00294) supervised by the IITP (Institute for Information & communications Technology Planning & Evaluation). *(Corresponding author: Eui-Nam Huh and Choong Seon Hong)*

VanDung Nguyen, Tran Trong Khanh, Thant Zin Oo, Eui-Nam Huh and Choong Seon Hong are with the Department of Computer Science and Engineering, Kyung Hee University, Yongin 446-701, South Korea (e-mail: ngvandung85@khu.ac.kr; khanhtrannnn@khu.ac.kr; tzoo@khu.ac.kr; johnhuh@khu.ac.kr; cshong@khu.ac.kr).

Nguyen H. Tran is with the School of Computer Science, The University of Sydney, Sydney, NSW 2006, Australia (e-mail: nguyen.tran@sydney.edu.au).

the scalability of the whole system, i.e., when a vast number of mobile devices and gadgets run a variety of application tasks simultaneously. To solve these challenges, a fuzzy logic–based MEO in [4] was discussed to solve the online problem by capturing the intuition of a real-world administrator to get an automated management system. Nevertheless, the number of failed tasks and the service time when the system is overloaded are still challenges because of WAN congestion.

Furthermore, IoT applications generally consist of several divisible applications, which are logically independent and require timely and context-aware processing. For example, computation-intensive components of an augmented reality application, namely, the tracker, mapper, and objective recognizer can be offloaded from IoT devices to MEC servers while video source and renderer are executed locally [9]. Mobile users often need to process a large amount of raw data from compute-intensive infotainment applications where the task involves compressing and uploading the data to the edge cloud for analysis and storage. Many works have considered the partial task offloading problem in MEC systems [9], [10], [11], [12]. These works assume that the base station (BS) has perfect knowledge of multi-user channel gains, local computing energy per bit, and input data size of all users, which can be obtained by feedback. Using this information, the BS selects offloading users, determines the offloaded data sizes with the criterion of minimum weighted sum mobile energy consumption [9], [10], [11], or minimum system delay of all users [12]. However, under unexpected variation of the load (e.g., the CPU utilization of a VM is frequent change depending on the tasks running on it), it is difficult to decide where the offloaded task should be processed. In this paper, by using the resource information and matching it with application requirements, the MEO will find the optimal task splitting ratio for the offloaded task.

The main contributions of this work can be summarized as follows. First, we propose a novel approach by integrating fuzzy logic and the optimal task-splitting strategy in the MEO to execute more time-critical small applications and reduce service times of delay-tolerant applications. Second, our approach can select the appropriate host for task execution. Third, by using the resource information and matching it with application requirements, our system decide mobile users to offload their tasks or a portion of the tasks to either an edge or a cloud server to minimize system delay for an incoming task through a collaboration between cloud computing and edge computing methods. Finally, we evaluate the performance of our proposal under the incoming task traffic of four application classes: augmented reality, healthcare, compute-intensive and infotainment applications.

II. System Model and Problem Formulation

In this paper, we focus on data partitioned oriented application models. The typical examples are the video compression [11], image processing, augmented reality [9], [10], [12], the virus scan, and the file/figure compression applications [13]. According to [9], [10], [11], [12], [13], we assume that each mobile user has a delay-sensitive application, which is composed of several separable tasks, e.g., face detection,

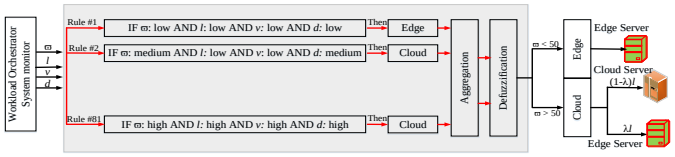

Fig. 2. Fuzzy logic architecture and the task-splitting strategy.

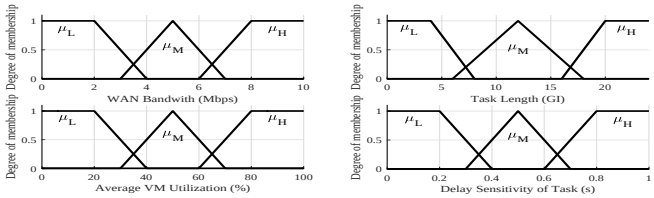

Fig. 3. Membership functions.

virus scanning and G-zip compression [12], [13]. The small applications are defined as they generate small tasks and medium-sized tasks, such as the healthcare and augmented reality applications. The advantage of parallelism is to concurrently process a portion of total data at the local side and the remainder at the cloud side. The critical function of the fuzzy logic–based mobile edge orchestrator is to find the target server (either a local edge server or a cloud server) for a task or to split the task for execution. Figure 2 shows the fuzzy inference system and task-splitting strategy for a workload orchestrator. This system is used to map different metrics into single values. The fuzzy inference system consists of a singleton fuzzifier, the product inference engine, the centroid defuzzifier, and a task-splitting scheme, as shown in Figure 2.

A. Fuzzy-based Workload Orchestrator

The fuzzy logic–based MEO is aware of the status of network resources and communications. This information is the input variables for operation of the fuzzy inference system in order to find a target server for an incoming task. According to [4], there are four input variables, given as:

$$\mathbf{F} = \{w, l, v, d\} \tag{1}$$

where w, l, v, d are WAN bandwidth, the length of incoming tasks, VM utilization of the edge server, and the delay sensitivity of related tasks, respectively. The task length, l, is used to compute the execution time of a task. Delay sensitivity of a task, d, shows the tolerance of the task to a longer execution time, either due to network conditions or due to server utilization levels [4]. Network and resource information is used to determine $\{w, v\}$. For WAN communication latency, network congestion is an important indicator in w as to when the task will be offloaded to the cloud server. VM utilization, v, gives information on the residual computational capacity of the edge server. If v is greater than a threshold level, the edge server is considered congested.

According to the four input variables, we set up four membership functions, which that are used in fuzzification and de-fuzzification steps. The linguistic variables of w, l, d are low (L), medium (M), and high (H). The linguistic variables of v are light (L), normal (N), and heavy (H). They are shown in Figure 3. Associate a grade to each linguistic term, and the crisp value is transformed into fuzzy values in the fuzzification step by using these membership functions. They are given as

$$\mathbf{F}_i(x) = [\mu_i^L(w), \mu_i^M(x), \mu_i^H(x)], \text{ where } i \in \{w, l, v, d\} \tag{2}$$

The inference step is the process of evaluating and combining fuzzy rules from the fuzzy rule base. A fuzzy variable, which is obtained after the inference step, is used in the defuzzification step. According to [4], a simple IF-THEN rule with a condition and a conclusion is used in the fuzzy rule base. In fuzzification, there are four membership functions with three linguistic terms; therefore, the number of fuzzy rules is $n = 3^4 = 81$. To determine the fuzzy rules, we vary the relatively better fuzzy rule set that is found empirically, and the best rule combination in the computational experiments is used [4]. Figure 2 shows example rules.

B. Delay Analysis

After we calculate the crisp output value from the fuzzy inference engine, the MEO will determine whether tasks should be processed by an edge server, or the edge server and a cloud server collaboratively. In edge server and cloud server collaboration, the task is divided into two parts that can be processed in the edge server and the cloud server as well. Each mobile device operates as follows.

1) Each mobile device will directly send its application profile to the connected edge node through the wireless channel. The MEO will obtain and update application information from a metropolitan area network (MAN).
2) By using the resource information and matching it with application requirements, the MEO will use Fuzzy based approach to select a target server. If the cloud server processes tasks, the MEO will find the optimal task splitting ratio for this incoming task.
3) Each mobile device, based on the decisions of the MEO, will offload tasks to either an edge or a cloud server.
4) The target MEC will serve devices for edge computing, while other parts of the task will be offloaded to the cloud server through the WAN.
5) The computation results are transmitted back to each mobile device.

1) Transmission delay of the mobile device: In accordance with the standard practice in computation offloading modeling [3], [14], the average transmission delay for the i^{th} user to offload a computation task to the connected j^{th} edge node is given as

$$t_{j,i}^{tran,u} = \frac{l_{j,i}T}{R} \tag{3}$$

where T represents the length of one time division multiple access (TDMA) frame, and R is the average data rate for the upload. Note that we use average data rate R to calculate the average transmission delay to match the time scale of task offloading. $l_{j,i}$ represents the task length, which is assumed to be an exponentially distributed random variable. Moreover, the size of the computation result is small enough so that the download delay can be ignored [12].

2) Computation delay of the edge server: Denote $\lambda_{j,i} \in [0, 1]$ the task-splitting ratio, which accounts for the data portion executed on the edge server [12], such that

$$t_{j,i}^{comp,e} = \frac{\lambda_{j,i} l_{j,i} \beta_{j,i}}{f^e} \tag{4}$$

which is the computation resources that the j^{th} edge node allocates to the i^{th} device in f^e giga-instructions (GI) per second (GIPS). $\beta_{j,i}$ is the number of CPU cycles required compute one-bit data of this task [12].

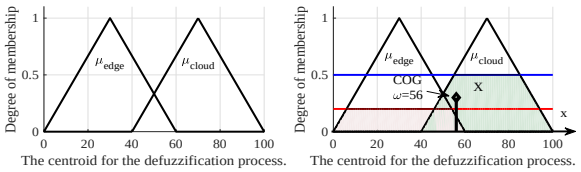

Fig. 4. The centroid for the defuzzification process.

3) Transmission delay of the edge cloud: In Figure 1, all edge nodes connect with the cloud server through a WAN, which covers a wide area from MEC servers to the cloud server in geography. Because the high-speed WANs have minimal transmission time and the data loss rate is not considered, for long distances of WANs, the transmission delay is dominated by the propagation time, defined as t^{WAN}.

4) Computation delay of the cloud server: When receiving an incoming task from an edge node, the MEO will allocate the available computation resources of the cloud server to each task for parallel computing by using the network information and matching it with the task requirements. Similar to f^e, f^c (in GIPS) represents the cloud computation resources processed for the i^{th} device served by the cloud server.

$$t_{j,i}^{comp,c} = \frac{(1 - \lambda_{j,i}) l_{j,i} \beta_{j,i}}{f^c} \tag{5}$$

III. Offloading Decision and Optimal Task-Splitting Strategy

A. Offloading decision

According to the number of fuzzy rules — i.e., Fig. 2 based on fuzzification function Eq. (2)— we use minimum (*min*) and maximum (*max*) functions to determine how the results of multiple rules are combined within a rule set [4].We calculate a fuzzy value for selecting the edge and cloud servers as follows:

$$\mu_{edge} = max\{\mu_{edge}^{R1}, ..., \mu_{edge}^{Rn}\} \tag{6}$$

$$\mu_{cloud} = max\{\mu_{cloud}^{R3}, ..., \mu_{cloud}^{Rn}\} \tag{7}$$

where the aggregation and activation phases are given as

$$\mu_{edge}^{R1} = min\{\mu_w^{R1}(x), \mu_l^{R1}(y), \mu_v^{R1}(z), \mu_d^{R1}(v)\} \tag{8}$$

$$\mu_{cloud}^{R3} = min\{\mu_w^{R3}(x), \mu_l^{R3}(y), \mu_v^{R3}(z), \mu_d^{R3}(v)\} \tag{9}$$

in which x, y, z, v are the crisp input parameters for a fuzzy inference system.

Based on $\{\mu_{edge}, \mu_{cloud}\}$, a crisp output value is calculated during defuzzification. In this step, we use a centroid defuzzifier, which returns the center of gravity (COG) of the area under the curve [4], as shown in Figure 4. It is calculated as

$$\omega = \frac{\int_{x \in X} x \mu_i(x)}{\int_{x \in X} \mu_i(x)}, i \in \{\text{edge, cloud}\} \tag{10}$$

According to crisp output ω, we make the offloading decision as follows. If ω is greater than 50, the task will be offloaded to the cloud server. Otherwise, it will be executed on the edge server.

B. Optimal task splitting strategy

According to Section II-B, the total delay of a task from the i^{th} device served is

$$T_{j,i} = t_{j,i}^{tran,u} + \max\{t_{j,i}^{comp,e}, t^{WAN} + t_{j,i}^{comp,c}\} \quad (11)$$

We aim at minimizing the delay of each task that is managed by the MEO. The optimization problem can be formulated as

$$\min_{\lambda_{j,i}} \quad T_{j,i} \quad (12a)$$

$$\text{s.t.} \quad \lambda_{j,i} l_{ij} \beta_{j,i} \leq B_j \quad (12b)$$

$$0 \leq \lambda_{j,i} \leq 1 \quad (12c)$$

where (12b) implies that the required computation resources of an incoming task should not exceed the maximum CPU resources provided on the VM by the j^{th} edge server, B_j. Note that, in a set of edge servers, the j^{th} edge server that has maximal CPU resources is denoted by $B_j^{\max}$. The optimized variable is the task-splitting ratio, $\lambda_{j,i}$.

Theorem 1. *The optimal task-splitting strategy, $\lambda_{j,i}^{opt}$, can be calculated as*

$$\lambda_{j,i}^{opt} = \begin{cases} \dfrac{f^e(f^c t^{WAN} + l_{j,i}\beta_{j,i})}{l_{j,i}\beta_{j,i}(f^e + f^c)}, & \text{if MEO looks up a edge server} \\ j^{th} \text{ satisfy } B_j \geq \dfrac{f^e(f^c t^{WAN} + l_{j,i}\beta_{j,i})}{(f^e + f^c)}. \\ B_j^{\max}/(l_{ij}\beta_{j,i}), & \text{otherwise.} \end{cases} \quad (13)$$

Proof. We have $t_{j,i}^{comp,e} = \dfrac{\lambda_{j,i} l_{j,i}\beta_{j,i}}{f^e}$ which increases with $\lambda_{j,i}$. Therefore, considering (12c), $\lambda_{j,i} \in [0,1]$, and we can derive $t_{j,i}^{comp,e} \in [0, \dfrac{l_{j,i}\beta_{j,i}}{f^e}]$. On the other hand, we have $t^{WAN} + t_{j,i}^{comp,c} = t^{WAN} + \dfrac{(1-\lambda_{j,i}) l_{j,i}\beta_{j,i}}{f^c}$, which decreases with $\lambda_{j,i}$. Thus, when $\lambda_{j,i} \in [0,1]$, we derive $t^{WAN} + t_{j,i}^{comp,c} \in [t^{WAN}, t^{WAN} + \dfrac{l_{j,i}\beta_{j,i}}{f^c}]$. Recall (11), and we find $\max\{t_{j,i}^{comp,e}, t^{WAN} + t_{j,i}^{comp,c}\}$; however, it first decreases and then increases with $\lambda_{j,i}$. We can find the intersection of two lines, given as $\lambda_{j,i}^* = \dfrac{f^e(f^c t^{WAN} + l_{j,i}\beta_{j,i})}{l_{j,i}\beta_{j,i}(f^e + f^c)}$. From (12b), we have $\lambda_{j,i} \leq B_j/(l_{ij}\beta_{j,i})$. Compare $\lambda_{j,i}$ and $\lambda_{j,i}^*$, and we can calculate $\lambda_{j,i}^{opt}$. As a result, the minimum value is achieved when $\lambda_{j,i} \geq \lambda_{j,i}^*$, which results in the optimal task-splitting ratio $\lambda_{j,i}^{opt} = \lambda_{j,i}^*$. Otherwise, $\lambda_{j,i} \geq \lambda_{j,i}^*$, and $\lambda_{j,i}^{opt}$ is $B_j/(l_{ij}\beta_{j,i})$. □

From the theorem in III-B, $\lambda_{j,i}^{opt}$ will become greater than 1 if the computation delay of the edge server is less than the transmission delay of the edge cloud, which means all tasks are associated with the edge node and processed in the edge server without offloading them to the cloud server. In following these reasons and considering a task offloaded to the cloud server, we present the necessary condition for a task to be offloaded to both edge and cloud servers.

Corollary 1. *If there exists an optimal task-splitting value, the task length must satisfy*

$$l_{j,i} > \frac{t^{WAN} f^e}{\beta_{j,i}} \quad (14)$$

Based on the optimal task-splitting ratio, $\lambda_{j,i}^{opt}$, the MEO will match it and the existing networking resources to select the appropriate edge server in order to satisfy the optimization constraints.

TABLE I: Application types used [4]

	AR	Healthcare	CI	Infotainment
Usage percentage (%)	30	20	20	30
Task interval (sec)	2	3	20	7
Delay sensitivity (%)	0.9	0.7	0.1	0.3
Active/idle period (sec)	40/20	45/90	60/120	30/45
Upload/download data (Kb)	1500/25	20/1250	2500/200	25/1000
Task length (GI)	9	3	45	15
VM utilization on edge (%)	6	2	30	10
VM utilization on cloud (%)	0.6	0.2	3	1

TABLE II: Simulation parameters [4]

Parameters	*Value*
Simulation time/warm-up period	33 /3 min
WAN/WLAN bandwidth	empirical
MAN bandwidth	MMPP/M/1 model
LAN propagation delay	5 ms
Number of VMs per edge/cloud server	8/4
Number of cores per edge/cloud VM CPU	2/4 min
VM CPU speed per Edge/Cloud	10/100 GIPS
Mobility model	Random way point
Propagation of selecting a location type	Equal
Number of location for Type 1/2/3	2/4/8
Mean dwell time for Type 1/2/3	2/5/8 min

IV. Simulation results

Aiming for real-world simulated models, we use four different application types: i) an augmented reality application on Google Glass, ii) infotainment application, iii) health application which uses a foot-mounted inertial sensor to analyze the walking pattern of the users, and iv) compute-intensive application; for more details, refer to EdgeCloudSim simulator [15], [4]. In our simulations, the mobile devices send these tasks to the remote servers, which provide related services, i.e., face recognition, fall risk detection, infotainment services. Different profiles for task arrival distribution, delay tolerance, and task size are shown in Table I [4]. The usage percentage of the application defines how the percentage of mobile devices running this application. We define how frequently the related task is sent to the edge orchestrator by task inter-arrival time, and it follows an exponential distribution. We assume that mobile devices generate tasks during the active period, and they just transmits in the idle period. Data is sent to/received from the server with the upload/download data rate. The delay sensitivity, task length, and VM utilization are used to determine the fuzzy inference system in Section II-A. Moreover, the simulation parameters are presented in Table II [15], [4]. We assume that a single server queue is modeled using Markov-modulated Poisson process (MMPP) arrivals [4]. When the state of the system congestion level is changed, the mean arrival rates of the tasks are updated. Therefore, an empirical study is carried out for characterizing the Internet connection capacity to measure the WLAN/WAN bandwidth [4].

For evaluation, we compare our proposal with three other approaches: i) using fuzzy logic without a collaborative cloud and edge scheme (*fuzzy*-based approach), ii) using the edge servers only if the CPU is not too high (*utilization*-based approach), and iii) finally, considering CPU and bandwidth (*hybrid* approach). The *fuzzy-co*–based approach splits each task for offloading to either an edge server or a cloud server. The task-splitting ratio is calculated based on the constraint between the required capacity and existing CPU resources on the VM . Figure 5a shows the average failed task ratios based

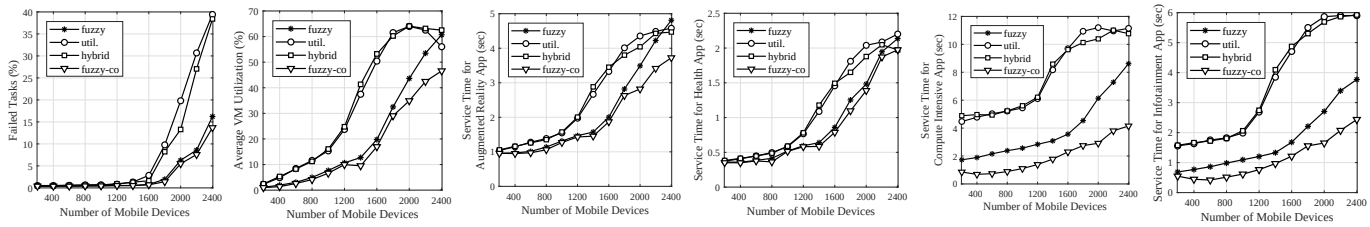

Fig. 5. Simulation results: a) Average failed-task ratio, b) Average VM utilization, c) Augmented reality application, d) Healthcare application, e) Compute-intensive application, and f) Infotainment application.

on all application types. Failed tasks are defined as i) tasks dropped by the network because of network congestion and ii) tasks failed due to insufficient CPU resources on the VM for the incoming task. By considering CPU resource conditions in the optimization constraints, our proposal provides the best performance.

In the IoT scenario, a large number of tasks that arrive at the same edge server or cloud server will make the congestion causing packet losses, exhaust resources, and facing scalability problems [16]. Using selective offloading and allocating resources can resolve the scalability problem [16]. The fuzzy-based MEO, based on the WAN bandwidth, VM utilization on the edge server makes the offloading decision for incoming tasks, i.e., if the WAN congestion is too high, offloading to the cloud server is ignored. Moreover, the task partitioning schemes in our proposal can help devices select the offloaded server among multiple edge servers. The effect on the scalability exploits the average CPU utilization of VMs running on the edge servers versus the number of mobile devices. Figure 5b shows when the number of task increases, our proposal utilizes the edge servers more efficiently than fuzzy, utilization, and hybrid approaches.

The healthcare and augmented reality applications require faster responsiveness (Table I). The *fuzzy-co*–based algorithm outperforms others when serving time-critical applications, such as the healthcare and the augmented reality applications, as shown in Figures 5c and 5d. Furthermore, the infotainment application generates big tasks, whereas the compute-intensive application generates very big tasks. Since the VMs running on cloud servers are very powerful in our scenario, the processing time is low. Hence, most compute-intensive and infotainment application tasks are executed on the cloud servers, providing better performance. In particular, the *fuzzy-co* algorithm can divide a task into two parts with one part processed on an edge node, and the other part offloaded to a cloud server to minimize the end-to-end latency for the mobile user, as shown in figures 5e and 5f. Therefore, by considering both processing delay and communication delay and optimizing latency by splitting the task, the *fuzzy-co* algorithm provides the lowest service times in all cases studied.

V. Conclusion

The main objective of this study is to improve service time and the failed-task ratio by integrating fuzzy-based workload orchestration and a collaborative edge and cloud computing scheme for latency minimization. A fuzzy-based MEO considers both computational and communication resources and makes the offloading decision. Moreover, we proposed an optimal task-splitting strategy for each task to achieve minimized service time. We further highlighted some insights into the task-splitting strategy by analyzing four realistic applications. Our proposal provides better results compared to other algorithms. Future work will collaborate with divisible load theory for a further-enhanced fuzzy-based workload orchestrator to reduce the completion time of IoT applications.

References

[1] V. D. Nguyen *et al.*, "Joint offloading and ieee 802.11p-based contention control in vehicular edge computing," *IEEE Wireless Communications Letters*, pp. 1–1, 2020.

[2] L. F. Bittencourt *et al.*, "Mobility-aware application scheduling in fog computing," *IEEE Cloud Computing*, vol. 4, no. 2, pp. 26–35, March 2017.

[3] A. Jaddoa *et al.*, "Dynamic decision support for resource offloading in heterogeneous internet of things environments," *Simulation Modelling Practice and Theory*, vol. 101, p. 102019, 2020, modeling and Simulation of Fog Computing.

[4] C. Sonmez *et al.*, "Fuzzy workload orchestration for edge computing," *IEEE Transactions on Network and Service Management*, vol. 16, no. 2, pp. 769–782, June 2019.

[5] V. Karagiannis *et al.*, "Network-integrated edge computing orchestrator for application placement," in *2017 13th International Conference on Network and Service Management (CNSM)*, Nov 2017, pp. 1–5.

[6] A. Hegyi *et al.*, "Application orchestration in mobile edge cloud: Placing of iot applications to the edge," in *2016 IEEE 1st International Workshops on Foundations and Applications of Self* Systems (FAS*W)*, Sep. 2016, pp. 230–235.

[7] A. C. Baktir *et al.*, "Enabling service-centric networks for cloudlets using sdn," in *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, May 2017, pp. 344–352.

[8] V. De Maio and I. Brandic, "First hop mobile offloading of dag computations," in *2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, May 2018, pp. 83–92.

[9] Y. Dai *et al.*, "Joint computation offloading and user association in multi-task mobile edge computing," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 12, pp. 12 313–12 325, 2018.

[10] C. You *et al.*, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Transactions on Wireless Communications*, vol. 16, no. 3, pp. 1397–1411, 2017.

[11] J. Ren *et al.*, "Latency optimization for resource allocation in mobile-edge computation offloading," *IEEE Transactions on Wireless Communications*, vol. 17, no. 8, pp. 5506–5519, 2018.

[12] J. Ren *et al.*, "Collaborative cloud and edge computing for latency minimization," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 5, pp. 5031–5044, May 2019.

[13] Y. Wang *et al.*, "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE Transactions on Communications*, vol. 64, no. 10, pp. 4268–4282, 2016.

[14] J. Niu *et al.*, "Bandwidth-adaptive partitioning for distributed execution optimization of mobile applications," *Journal of Network and Computer Applications*, vol. 37, pp. 334 – 347, 2014.

[15] C. Sonmez *et al.*, "Edgecloudsim: An environment for performance evaluation of edge computing systems," *Transactions on Emerging Telecommunications Technologies*, vol. 29, no. 11, p. e3493, 2018, e3493 ett.3493.

[16] X. Lyu *et al.*, "Selective offloading in mobile edge computing for the green internet of things," *IEEE Network*, vol. 32, no. 1, pp. 54–60, 2018.