Group-Delay Aware Task Offloading with Service Replication for Scalable Mobile Edge Computing

Shimaa A. Mohamed*†, Sameh Sorour*, and Hossam S. Hassanein*
* School of Computing, Queen's University, Kingston, ON, Canada
† Information Technology Institute, City of Scientific Research and Applications, Alexandria, Egypt
Email: {16saa4, sameh.sorour, hassanh}@queensu.ca

GLOBECOM 2020 - 2020 IEEE Global Communications Conference | 978-1-7281-8298-8/20/$31.00 ©2020 IEEE | DOI: 10.1109/GLOBECOM42002.2020.9348241

***Abstract*—A rapid increase has been lately noticed in the number of individual and groups of users offloading independent and inter-related computational tasks to mobile edge computing (MEC) servers, thus overloading them and increasing risks of service interruptions. In response to this issue, reactive service replication has been suggested to enable individual and groups of users to access services on remote edge servers, thus guaranteeing system scalability. In this paper, we propose a task offloading and service replication scheme on local and remote MEC servers, which minimizes the response time of all users while satisfying the delay requirements of user groups involved in same traffic-heavy and/or multimedia-intense applications (e.g., online gaming, multimedia conferencing, augmenting reality). We formulate the problem as an integer non-linear problem, and solve it using numerical solvers. We then compare the performance of our optimized solution with distance-based and resource-based greedy approaches. Simulation results show that our optimized solution can achieve up to 14% and 13% performance gains in comparison to these two greedy approaches, respectively.**

***Index Terms*—Mobile Edge Computing, Computation Offloading, Service Replication**

I. Introduction

Due to the resource limitations of mobile devices (processing power, battery lifetime, and storage capacity), many techniques have been proposed to handle the complex computations currently needed by such devices. One of the conventional solutions to this problem is computation offloading, where part or all such computations are offloaded to remote resourceful cloud servers, thus saving processing power and energy. However, offloading these computations to the cloud may cause long communication latency between cloud servers and mobile devices. Mobile Edge Computing (MEC) has thus emerged as an alternative technology that enables mobile devices to access cloud-like computing services at edge servers located within the radio access network of the mobile subscribers. The major objective of MEC is reducing latency by enabling the computations and data storage to be done at these edge servers [1]. Despite their significant potential to improve the performance of delay-sensitive applications, edge servers have relatively limited computation resources compared to cloud servers. This fact may cause edge servers to be overloaded with computations to the extent of service interruption [1]. This is especially true in scenarios of high density demand such as those sports events and live concerts.

To minimize such instances, partial and full task offloading and resource allocation strategies have been extensively investigated by many researchers either in single-server or multiple servers edge computing environments. In [2], Du et al. proposed a joint task offloading and resource allocation scheme for multi-server edge computing environments. This scheme intended to maximize service capacity (defined by the number of served mobile devices) and minimize the service cost (measuring the service latency and power consumption experienced by users). However, this work assumed that some users can be left unserved due to lack of resources. On the other hand, another line of research aimed to jointly optimize task allocation and resource scheduling to reduce the cost of MEC while assuming that the capacity of the edge servers will always satisfy the demands of the users. In [3], Zhang et al. investigated the joint task scheduling and dynamic resource management problem to reduce the cost of an edge computing system consisting of multiple collocated edge servers. In [4], Li et al. proposed a mixed-integer non-convex optimization problem to maximize the number of offloaded tasks and minimize the energy consumption of all the user devices and edge servers, by selecting the best edge servers with capacity for offloading.

The majority of the above works ignored the notion of scalability in MEC, which guarantees the availability of service regardless of the number of user devices and capacities of the edge servers. With the increasing number of devices and complexity of the applications that run on them, a massive

978-1-7281-8298-8/20/$31.00 ©2020 IEEE

load is typically imposed on the local edge server from these devices resulting in network bottlenecks and service interruptions [1]. In [5], Maheshwari et al. analyzed the scalability and capacity performance of a hybrid edge-cloud system designed to support latency-sensitive applications. On the other side of the spectrum, the distribution of offloaded tasks between edge servers and peer user devices was proposed to determine the best load partitioning between them for minimizing the excess delay and energy consumption compared to the no-offloading alternative [6].

To achieve scalability within the edge servers' layer itself, many researchers have suggested load balancing mechanisms and server clustering to ensure the scalability of the service. In [7], a first-come-first-serve queuing and service model was proposed to improve the QoS of edge servers by clustering these servers together for load sharing. In [8], a cooperative load balancing scheme was proposed to minimize the blocking states at each edge server. In [9], collaborative data caching and computation offloading among collaborating MEC servers were proposed. The caching and computation resources are allocated to multiple service requesters based on their demands and payments.

The difference between this work and the aforementioned ones is that our problem considers multiple sets of users forming service groups due to the participation of each group in the same synchronized traffic-heavy and/or multimedia-enabled activities, such as online gaming, multimedia conferencing, and augmented reality. Also, our proposed scheme improves the scalability of the offered services by replicating them on-demand at remote edge servers in the MEC system to serve all users concurrently when the local server(s) cannot bare the load of all user groups.

In this paper, we dynamically manage task offloading for groups of users requesting to perform inter-related computations on local and remote MEC servers. We formulate an optimization problem minimizing the response time of all users while satisfying the delay requirements of user groups running the same applications. We compare the performance of our optimized solution with distance-based and resource-based greedy approaches.

The rest of the paper is organized as follows, Section II details our system model and parameters. Section III discusses the problem setup. Section IV presents the performance evaluation and the comparison with the two greedy approaches. We conclude the paper and highlight future research directions in Section V.

II. System Model

The system model is depicted in Figure 1, in which the MEC system consists of two sets of entities namely the set of edge servers and the set of users requesting service from these servers. Let $\mathcal{S} = \{0, 1, 2, \ldots, M-1\}$ be the set of M edge servers in the system, where edge server 0 is the local edge server and edge server $i \in \mathcal{S} \setminus \{0\}$ is one of the $M-1$ remote edge servers. The local server is the one that all users should typically offload to unless overloaded. Only in this case, the un-served devices are allowed to access remote edge servers after replicating their needed service(s) on them.

Edge servers are connected via backhaul links, which can be used for service replications whenever needed. In addition, their computational resources are divisible among multiple users from the same or different groups. The available computing capacity of each edge server $i \in \mathcal{S}$ is denoted by F_i measured in gigahertz (Ghz).

On the user side, $\mathcal{U} = \{1, 2, \ldots, N\}$ denotes the set of all N user devices in the MEC system. These users are split into a set $\mathcal{G} = \{1, 2, \ldots, NG\}$ of NG groups. Each group $g \in \mathcal{G}$ consists of U_g users, where users belonging to the same group are participating in the same common traffic-heavy and/or multimedia application/activity, such as online gaming, multimedia conferencing, and augmented reality.[1] Consequently, each group has application-dependent QoS requirements T_g^{max} and F_g^{min}, defined as the maximum delay all members within the group can tolerate for having their tasks completed and the minimum CPU speed these tasks can be processed with, respectively. We assume that users can all connect to the local edge server, which is typically collocated with their nearest base station (BS) usually identified according to its received signal strength. Whenever needed, each user can also wirelessly connect to one of a subset or all the $M-1$ remote servers with different levels of connection qualities, which are impacted by to its distance and fading conditions with each of them.

At execution time, each user sends its task to one of its accessible edge servers. A user's task T_u is identified as:

$$\mathcal{T}_u = (C_u, B_u, g), \quad \forall u \in \mathcal{U}, \tag{1}$$

where C_u (measured in gigacycle) denotes the total number of the CPU cycles required to complete user u's task, B_u (measured in bits) describes the size of the information that user u needs to send to the edge server for it to execute this task, and g denotes the ID of the group user u belongs to.

[1]This model captures as a special case scenarios where any one or multiple users are solely involved in applications. In this case, each of these users forms a separate group of size 1 (i.e., $U_g = 1$).

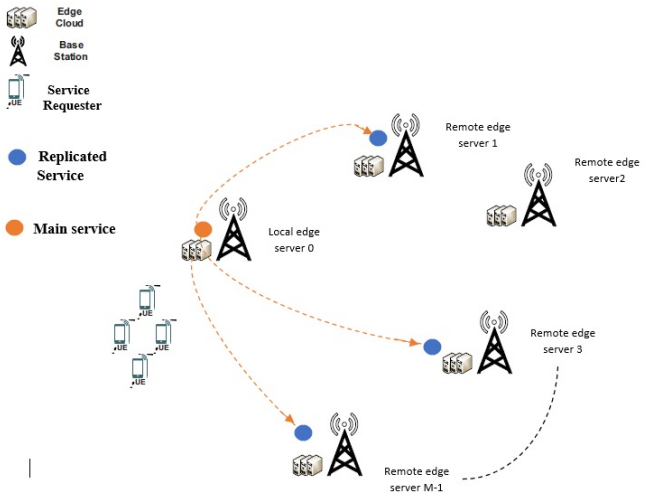

Fig. 1. System Model

III. Problem Setup

In this section, we formulate our group-delay aware task offloading and service replication problem with the aim to minimize the total response time of all users served by both local and remote edge servers, while satisfying the delay requirements of user groups running the same applications.

A. Definition of Objective Variables

Let a_i, $i \in \mathcal{S}$ be a binary variable determining whether remote edge server i in the MEC system is active or not in serving our considered set of users. This variable is defined for $i \in \mathcal{S} \setminus \{0\}$ as:

$$a_i = \begin{cases} 1, & \text{Remote edge server } i \text{ is active} \\ 0 & \text{Otherwise.} \end{cases}$$

Clearly, $a_0 = 1$ is as the local edge server is always active and achieves the least time in performing the computations of a subset or all the user. In addition, let x_{ui}, $u \in \mathcal{U}$ and $i \in \mathcal{S}$ be task offloading indicator, where x_{ui} is set to 1 when user u offload its task to edge server i, i.e., $x_{ui} = 1$, and is set to 0 otherwise.

As in typical MEC settings, users offload their tasks and access services at their assigned edge server through the direct wireless channel between them. The rate at which the instructions and data of user u's computational task (i.e., $\mathcal{T}_u$) is sent to edge server i is thus expressed as [4],

$$R_{ui} = B \log_2 \left(1 + \frac{P_u H_{ui}}{\sigma_i B}\right) \quad \forall\, u \in \mathcal{U}, \forall\, i \in \mathcal{S} \tag{2}$$

where B is channel bandwidth, P_u is the transmission power of user u, σ_i denotes the noise power spectral density at edge server i, and H_{ui} denotes the channel power gain between user u to edge server i, defined as:

$$H_{ui} = d_{ui}^{-\alpha} \quad \forall\, u \in \mathcal{U}, \forall\, i \in \mathcal{S} \tag{3}$$

where d_{ui} is the distance between user u and edge server i , and α is the path loss exponent, which is typically set to 4 in mobile edge environments [10].

Consequently, the transmission time of Task $\mathcal{T}_u$ of user u to edge server i can be expressed as:

$$T_{ui}^{T} = \frac{B_u}{R_{ui}} \quad \forall\, u \in \mathcal{U}, \forall\, i \in \mathcal{S} \tag{4}$$

In addition, the computation time of task $\mathcal{T}_u$ at edge server i can be expressed as:

$$T_{ui}^{C} = \frac{C_u}{F_{ui}}. \tag{5}$$

where $F_{ui} \geq F_g^{min}$ is the assigned CPU frequency assigned to the task of user u that belongs to group g if it is offloaded to edge server i. Finally, if the task of user u is offloaded to remote edge server $i \in \mathcal{S} \setminus \{0\}$, an additional time T_{ui}^{R} will be required to replicate and deploy this service on that remote edge server. Clearly, T_{u0}^{R} is assumed to be 0 $\forall\, u \in \mathcal{U}$ as the local edge server has all the services ready to directly run for the tasks of its own users.Thus, the response time of the user u's task $\mathcal{T}_u$ at the edge server $i \in \mathcal{S}$ is given as,

$$T_{ui}^{resp} = T_{ui}^{T} + T_{ui}^{C} + T_{ui}^{R} \tag{6}$$

B. Problem Formulation

Given the above definition of objective variables, the tasks offloading process achieving our target can be expressed as follows:

$$\min_{\substack{a_i, x_{ui} \\ \forall\ i \in \mathcal{S}, u \in \mathcal{U}}} \ \frac{1}{N} \sum_{i=0}^{M-1} \sum_{u=1}^{N} a_i x_{ui} T_{ui}^{resp} \tag{7a}$$

$$S.T. \quad a_i \in \{0,1\} \qquad \forall\, i \in \mathcal{S} \setminus \{0\} \tag{7b}$$

$$a_0 = 1 \tag{7c}$$

$$x_{ui} \in \{0,1\} \qquad \forall i \in \mathcal{S}, u \in \mathcal{U} \tag{7d}$$

$$\sum_{i=0}^{M-1} x_{ui} = 1 \qquad \forall\, u \in \mathcal{U} \tag{7e}$$

$$\sum_{i=0}^{M-1} a_i x_{ui} = 1 \qquad \forall\, u \in \mathcal{U} \tag{7f}$$

$$a_i - \sum_{u=1}^{N} x_{ui} \leq 0 \qquad \forall\, i \in \mathcal{S} \setminus \{0\} \tag{7g}$$

$$\sum_{u=1}^{N} a_i x_{ui} F_{ui} \leq F_i \qquad \forall\, i \in \mathcal{S} \tag{7h}$$

$$\sum_{i=0}^{M-1} a_i x_{ui} T_{ui}^{resp} \leq T_g^{max} \qquad \forall\, g \in \mathcal{G}, u \in \mathcal{U} \tag{7i}$$

Clearly, the objective function in (7a) minimizes the average response time of all users tasks by determining the best edge server and service replication decisions. Constraints (7b) and (7d) impose binary decision values for the variables a_i of remote edge servers and x_{ui} of all users and servers. Constraints (7e) ensure that each task is offloaded to only one server, whereas Constraints (7f) ensure that an active server is handling each of these offloaded tasks (i.e., ensuring that $a_i = 1$ if $x_{ui} = 1$, thus preventing cases where $a_i = 0$ when $x_{ui} = 1$ for at least one u index). On the other hand, Constraints (7g) ensures that a remote edge server is never active unless at least one task is offloaded to it (i.e., ensuring $a_i = 1$ only if $x_{ui} = 1$ for at least one u index, thus preventing cases of $a_i = 1$ when all $x_{ui} = 0$). Constraints (7h) ensure that the sum of assigned CPU frequencies to all users offloading their tasks to each edge server does not exceeding its total available CPU frequency. Finally, Constraints (7i) ensure that the total response time of a user's task belonging to any group $g \in \mathcal{G}$ does not exceed this group's maximum delay requirement.

It can be easily inferred from (7) that the problem is an integer non-linear problem, which is known to be very complex to solve analytically. Yet, we will solve this problem using a numerical solver in this paper to assess its gains compared to the two possible greedy approaches that can satisfy the scalability and group-delay awareness settings considered in this paper.

IV. Performance Evaluation

A. Simulation Setup

Here, we present the simulation results to show the performance and merits of the considered problem's optimal solution. The physical and task-related parameters considered in our simulation are listed in Table I. The edge servers are deployed at random locations in the simulation area randomly. All groups of user devices are then deployed near to (yet at random distances from) one specific edge server, when is thus designated as the local edge server (i.e., element 0 in set $\mathcal{S}$).

The numerical solver employed to derive the optimized solution is the MATLAB Opti-Optimization Toolbox [11]. This solution's merits is then compared to two greedy approaches, namely (1) Distance-based approach in which users closes to the local servers offload their tasks to it and the remaining users offload their tasks to their closest available remote servers (i.e., the closest ones to them that can still satisfy both of their group's QoS requirements after admitting their closer users). (2) Resource-based approach in which users offload tasks first to the local server then to remote servers in descending order of their assigned CPU frequencies to them.

TABLE I
The Simulation Parameters

Parameter Name	Parameter Value (unit)
Simulation Area	3000×3000(m)
B	10^6(Hz)
F_0	100(Ghz)
$F_i, i \in \mathcal{S} \setminus \{0\}$	$[10-50]$(Ghz)
R_u	$[1-10]$(Gigacycle)
B_u	$[10-40]$(KByte)
The QoS Requirement T_g^{max}	$\{2, 2.2, 2.4, 2.6\}$(sec)
P_u	0.5(W)
σ	10^{-3}(W/Hz)

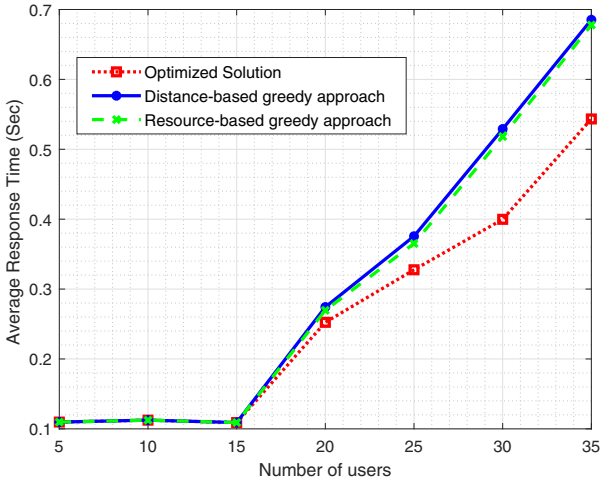

Fig. 2. Average response time vs. the number of users where the number of groups of user devices=1.

It is important to note that two described greedy approaches have exhibited cases where they were not able to serve all users. The merit evaluation is thus done based on two average parameters (where averages are derived over a very large number of trials with different server and user locations): (1) The average response time of all served users. (2) Average number of un-served users.

B. Simulation Results

Figures 2, and 3 show that the optimized solution outperforms the two greedy approaches in average response time as the number of users increased. In Figure 2, the optimized solution can achieve up to 14% and 13% response time reduction over the solutions achieved by distance-based greedy approach and resource-based greedy approach respec-

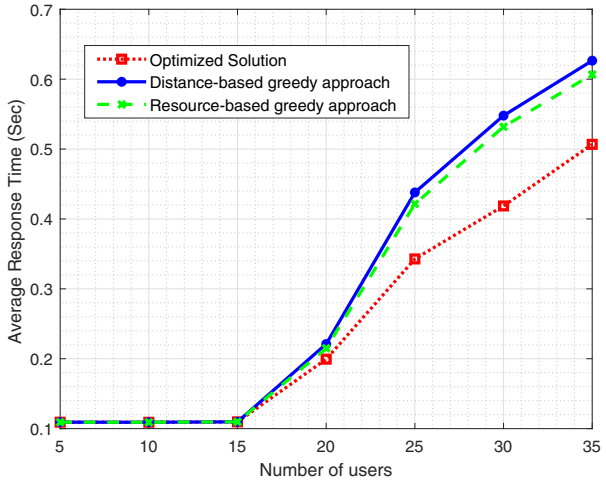

Fig. 3. Average response time vs. the number of users where the number of groups of user devices=3.

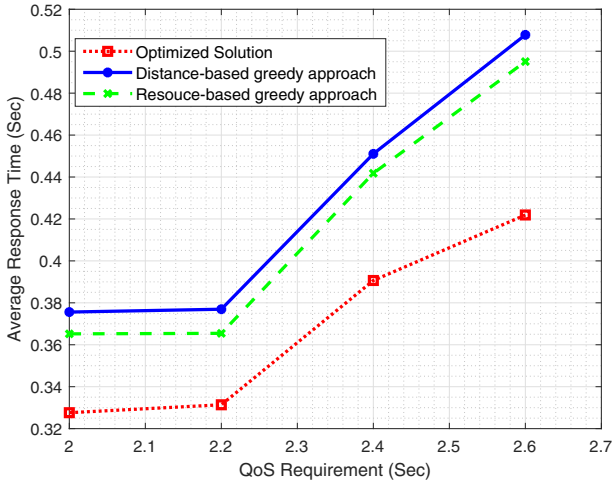

Fig. 4. Average response time Vs QoS requirement where the number of groups of user devices=1

tively when the number of users equals 35, where the number of the edge servers is set to 7 and the QoS requirement of the group equals 2 Sec, and the optimized solution can achieve up to 12% and 10% response time reduction over the solutions achieved by distance-based greedy approach and resource-based greedy approach respectively when the number of users equals 35, the number of the edge servers is set to 7 and the QoS requirement of the three groups equal $1.8, 2$, and 2.2 Sec as shown in Figure 3.

With different values of group's QoS requirements, the number of users is set to 25 and the number of edge clouds equals 7; the optimized solution can achieve up to 9% response time reduction compared with the above mentioned greedy approaches when all users belong to the same group as shown in Figure 4. While for multiple groups, the QoS of three groups are set to QoS, 0.9*QoS, and 1.1*QoS. The optimized solution can achieve up to 6% response time reduction compared with two greedy approaches as shown in Figure 5.

As the QoS requirement increased the difference of the average response time between the optimized solution and two greedy approaches increased, in case of $NG = 1$, from 0.04 Sec to 0.09 Sec approximately. While the case of $NG = 3$, the difference of the average response time between the optimized solution and two greedy approaches is nearly constant which at around 0.04 Sec and 0.06 Sec, respectively.

Additionally, when the range of required CPU cycles is increased, our optimized solution makes edge servers serve all users simultaneously while the two greedy approaches cannot

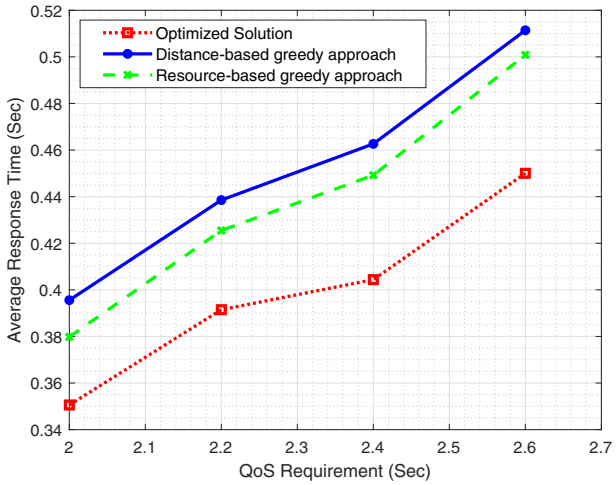

Fig. 5. Average response time Vs QoS requirement where the number of groups of user devices=3.

make edge servers serve all users. As shown in Figure 6, user tasks computing resources increased up to 30 gigacycle, the number of the edge servers is set to 7 and the QoS requirement of three group are set to $2.2, 2.4$, and 2.6 sec, the optimized solution can assign all user tasks at edge servers and edge servers introduce service to all users. On the other hand, two greedy approaches failed to assign all user tasks at edge servers, where the average number of unserved users increased from 20% to 30% when the number of users increased from 15 to 20 users.

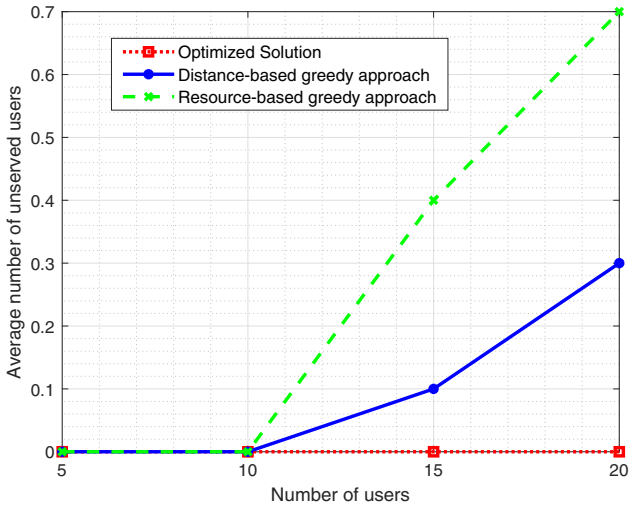

Fig. 6. Average number of users who are unserved vs. the number of users where the number of groups of user devices=3.

V. Conclusion

In this paper, we investigated the problem of task offloading with service replications in scalable MEC environments for groups of users involved in similar multimedia-intense applications. By scalable MEC, we mean MEC systems having the ability to replicate services on one or multiple remote servers, when the local server is overloaded, to avoid service interruption and accept the offloading requests of as many users as possible. The aim of our study was to minimize the average response time of all users' offloaded tasks, while respecting the application-imposed delay and QoS requirements on all the users of each group. The problem was formulated as an integer non-linear program and was then solved numerically using MATLAB's Opti-Optimization Toolbox. Simulation results show that this optimized solution outperforms the distance- and resource- based greedy schemes that respect our model's scalability and group service settings. More gains were achievable by our scheme as the number of users increases. The results also show that the average response time of the optimized solution reduced by up to 14% and 13% compared to both the distance- and resource-based greedy approaches.

In future works, we plan to consider scenarios in which service replication yield costs on the users, and extend our study to jointly minimize the offloading response time and cost We also aim to investigate scenarios in which users simultaneously access more than one service from the MEC system.

References

[1] Arif Ahmed and Ejaz Ahmed, "A Survey on Mobile Edge Computing," In 10th International Conference on Intelligent Systems and Control (ISCO), IEEE, 2016, pp. 1–8.

[2] Wei Du, Tao Lei, Qiang He, Wei Liu, Qiwang Lei, Hailiang Zhao, and Wei Wang, "Service Capacity Enhanced Task Offloading and Resource Allocation in Multi-Server Edge Computing Environment," arXiv preprint arXiv:1903.04709 (2019).

[3] Yongchao Zhang, Xin Chen, Ying Chen, Zhuo Li, and Jiwei Huang, "Cost efficient scheduling for delay-sensitive tasks in edge computing system," In 2018 IEEE International Conference on Services Computing (SCC), IEEE, pp. 73–80.

[4] P. Li, Y. Luo, K. Wang, and K. Yang, "Energy Minimization and Offloading Number Maximization in Wireless Mobile Edge Computing," In 2018 IEEE Global Communications Conference (GLOBECOM),IEEE, pp. 1–6.

[5] Sumit Maheshwari, Dipankar Raychaudhuri, Ivan Seskar, and Francesco Bronzino, "Scalability and performance evaluation of edge cloud systems for latency constrained applications," In 2018 IEEE/ACM Symposium on Edge Computing (SEC), IEEE, pp. 286–299.

[6] U. Yaqub, and S. Sorour, "Multi-Objective Resource Optimization for Hierarchical Mobile Edge Computing," In 2018 IEEE Global Communications Conference (GLOBECOM'18), Abu Dhabi, UAE, December 2018.

[7] L. Liu, S. Chan, G. Han, M. Guizani, and M. Bandai,"Performance modeling of representative load sharing schemes for clustered servers in multiaccess edge computing," IEEE Internet of Things Journal (2018), 6(3), pp. 4880–4888.

[8] R. Beraldi, A. Mtibaa, and H. Alnuweiri, "Cooperative load balancing scheme for edge computing resources," In 2017 Second International Conference on Fog and Mobile Edge Computing (FMEC), IEEE, pp. 94–100.

[9] A. Ndikumana, S. Ullah, T. LeAnh, N. H. Tran and C. S. Hong, . (2017, September), "Collaborative cache allocation and computation offloading in mobile edge computing," In 2017 19th Asia-Pacific Network Operations and Management Symposium (APNOMS),IEEE, pp. 366–369.

[10] L. Yuchong, W. Jigang, W. Yalan, and C. Long, "Task Scheduling in Mobile Edge Computing with Stochastic Requests and M/M/1 Servers," In 2019 IEEE 21st International Conference on High Performance Computing and Communications, IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), IEEE, pp. 2379–2382.

[11] https://inverseproblem.co.nz/OPTI/index.php/Probs/MINLP.