LETTER

Delay Improvement in Hierarchical Multi-Access Edge Computing Networks

Ngoc-Tan NGUYEN[†], *Member*, **Trung-Duc NGUYEN**[†], **Nam-Hoang NGUYEN**[†a)], *and* **Trong-Minh HOANG**[††], *Nonmembers*

SUMMARY Multi-access edge computing (MEC) is an emerging technology of 5G and beyond mobile networks which deploys computation services at edge servers for reducing service delay. However, edge servers may have not enough computation capabilities to satisfy the delay requirement of services. Thus, heavy computation tasks need to be offloaded to other MEC servers. In this paper, we propose an offloading solution, called optimal delay offloading (ODO) solution, that can guarantee service delay requirements. Specificially, this method exploits an estimation of queuing delay among MEC servers to find a proper offloading server with the lowest service delay to offload the computation task. Simulation results have proved that the proposed ODO method outperforms the conventional methods, i.e., the non-offloading and the energy-efficient offloading [10] methods (up to 1.6 times) in terms of guaranteeing the service delay under a threshold.

key words: *MEC, service delay, offloading, server selection*

1. Introduction

Multi-access edge computing (MEC) is one of the advanced technologies in 5G mobile networks because it is considered as the convergence of IT and telecommunications networking [1]–[3]. MEC provides the feature of cloud-computing to users by integrating with the radio access network (RAN) [1], [2]. By that way, it helps to solve the existing problem of cloud computing, i.e., service delay [3], [4].

With the feature of MEC, its architecture can be deployed in IoT systems where computation offloading is provided from IoT devices to edge servers. Consequently, IoT devices cannot only reduce processing delay and energy consumption, but also increase quality of service and lifetime. However, the deployment of MEC systems also have technical challenges in terms of resource allocation, scalability, security, user mobility, etc. [5], when providing computation offloading. Recent research activities on computation offloading in MEC systems have focused on server selection for offloading to enhance energy efficiency of users [6]–[8]. In particular, the authors in [6] study a minimization problem of energy consumption while ensuring the latency constraint by studying the energy cost of both computing and transmissionss tasks. A competitive game model is proposed in [7] which allows users to minimize its energy consumption subject to real-time constraints. The work [8] presents a minimization problem of the weighted sum energy consumption subject to the computation latency constraint. However, all of the aforementioned works deploy a single-layer MEC architecture that is no longer efficient with high offloading load. The authors in [10] propose an efficient server selection solution for the hierarchical MEC architecture which can optimize the energy consumption. Yet this work cannot guarantee the delay requirements under high system load.

In this paper, we first consider deploying a practical hierarchical MEC architecture [4], [9], [10]. We then propose an optimal delay offloading solution to support the guarantee of service delay. In particular, the queuing time of MEC servers is regularly updated and used to select an offloading server which is able to provide the minimum service delay.

2. System Model

The hierarchical MEC network model illustrated in Fig. 1 includes three layers of servers where 1st layer servers have point-to-point links to 2nd layer servers. The 2nd and 3rd layer servers are connected in a mesh topology. MEC servers at different layers have different computation capabilities depending on their deployment layer and performance [10]. High performance servers are deployed at the core network (the 3rd layer) whereas lower capacity servers are deployed at the radio access networks (the 1st layer). In the scope of this paper, we consider the system deployment scenario with fixed users located in the coverage of their access MEC server for a long duration, i.e., no mobility.

A computation task is represented by a tuple [10], i.e., $T = \{r^C, r^N, \tau^M\}$, where r^C is the computing resource required for performing the computation task, e.g., CPU cycles. r^N denotes the size of computation task measured in megabits, while τ^M is the maximum allowable service delay of the computation task measured in milliseconds. To represents the complexity of the computation task and its computing resource requirement, a parameter should be considered, called Computational Intensity (CI), which is the number of CPU cycles per computation unit. With a given CI, the required computing resource for a computation task is estimated as $r^C = r^N \mathrm{CI}$. Service delay of performing a computation task includes the computation time, transmissions time, and queuing delay [10]. Note that the transmissions time between the users and the 1st layer servers can be ignored because they are typically deployed close to each other. The computation

Manuscript received June 11, 2023.
Manuscript revised September 14, 2023.
Manuscript publicized March 5, 2024.

[†]VNU-University of Engineering and Technology, Vietnam.
[††]Posts and Telecommunications Institute of Technology (PTIT), Vietnam.
a) E-mail: hoangnn@vnu.edu.vn (Corresponding author)

DOI: 10.1587/transfun.2023EAL2048

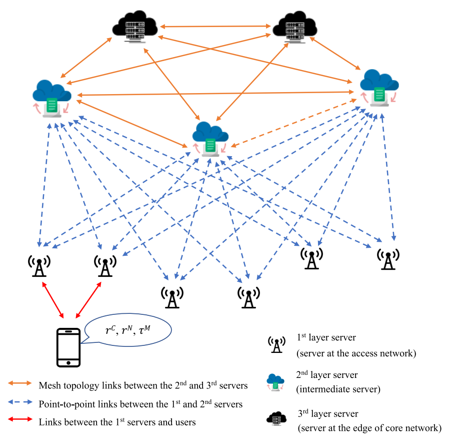
r^C, r^N, τ^M
1st layer server
(server at the access network)
2nd layer server
(intermediate server)
3rd layer server
(server at the edge of core network)
Mesh topology links between the 2nd and 3rd servers
Point-to-point links between the 1st and 2nd servers
Links between the 1st servers and users

Fig. 1 System model of hierarchical MEC networks.

time of a task at server j can be calculated by:

$$t_j^{\text{comp}} = \frac{r^C}{x_j^C}, \tag{1}$$

where x_j^C is the compute resource of server j allocated for a computation task, which is measured in the number of CPU cycles per second (CPU cycles/s). The transmissions time from server i to server j is calculated as follows:

$$t_{i,j}^{\text{trans}} = \frac{r^N}{x_{i,j}^N}. \tag{2}$$

Assume that each server only performs one computation task at a time, the queuing model M/M/1 is applied in our proposed system. The system load is $\rho_j = \lambda_j/\mu_j$, where λ is the arrival rate of computing tasks to server j. Meanwhile, the average speed of processing is expressed as follows:

$$\mu_j = \frac{C_j^{\text{CPU}}}{\bar{r}^N \text{CI}}, \tag{3}$$

where C^{CPU} is the speed of the server's CPU in cycles per second. CI denotes the computation intensity in cycles per bit, while $\bar{r}^N$ is the average size of computing tasks in bits. Thus, the queuing delay at server j can be estimated by:

$$t_j^{\text{queue}} = \frac{\rho_j}{\mu_j - \lambda_j} = \frac{\rho_j}{\mu_j(1-\rho_j)}. \tag{4}$$

From Eqs. (1), (2), and (4), the total service delay to complete an offloading task of server j can be expressed by:

$$t_j^{\text{total}} = 2.t_{i,j}^{\text{trans}} + t_j^{\text{comp}} + t_j^{\text{queue}}. \tag{5}$$

3. Optimal Delay Offloading Approach

In this section, we propose an optimal delay offloading (ODO) solution which optimizes both the transmissions delay and queuing delay by selecting a proper server to guarantee the service delay requirement. The hierarchical MEC network is modeled as a graph $G(s, e)$ where s is the set of servers and e is the set of connections between servers. The cost of each connection e_{ij} between server i and server j is defined as the transmissions delay. Each server s_j has the cost calculated as the sum of the queuing delay and computing time of the requested task. Assume that the information about the queuing delay of each server is estimated and update regularly to the access server. Note that each computing task requires a specific delay, i.e., τ^M. Thus, the selected offloading server j must satisfy the following constraint:

$$t_j^{\text{total}} < \tau^M. \tag{6}$$

In the case that the access server cannot ensure the service delay requirement, it offloads its computing tasks to a server at higher layers. The detail of the server selection offloading approach is summarized as follows:

1. **Access server selection (at users)**
   - Step 1: A user firstly measures signals of broadcast channels of nearby base stations to make a list of available access servers that it can connect to.
   - Step 2: It then selects one server in the 1st layer that has the strongest signal to be its access server.
2. **Offloading server selection (at access servers)**
   - Step 1: The access server firstly receives a computing request with the delay requirement (τ^M) from the user, then makes a decision whether selecting an offloading server to tackle the request.
   - Step 2: It then implements the Dijkstra algorithm to find an offloading server satisfying the condition that the total delay (t^{total}) of offloading the user's request to the offloading server is minimum.
   - Step 3: It validates the total delay whether satisfies the constraint (6). If the constraint (6) is satisfied, the offloading procedure is implemented. Otherwise, it sends a reject response to the user.

To evaluate the system performance of our proposed solution, we leverage the energy-efficient offloading solution proposed in [10] for performance comparison. In this approach, the system energy consumption, which is caused by transmissions and computation, is taken as the optimization objective of the offloading problem.

4. Numerical Results

In this section, numerical simulations are provided to evaluate the performance of the optimal delay offloading (ODO) method for the multi-tier MEC networks. We consider a simulation area of 1000m $\times$ 1000m with 1000 users distributed homogeneously. There are 9 access servers which is distributed homogeneously in the 1st-layer. The size of computing tasks is generated randomly in an exponential distribution with the mean of 1 Mbits. Bandwidth of links between the 1st and 2nd layers, and between the 2nd and 3rd

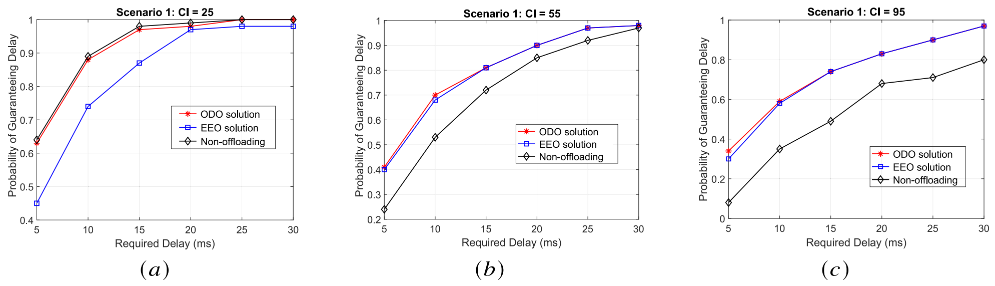

Fig. 2 Probability of guaranteeing service delay as (a) CI = 25, (b) CI = 55, and (c) CI = 95 in the case of low system load.

Table 1 Percentage of processed requests.

CI Levels	1st layer		2nd layer		3rd layer	
	EEO solution	**ODO solution**	**EEO solution**	**ODO solution**	**EEO solution**	**ODO solution**
CI = 25	0%	86%	0%	14%	100%	0%
CI = 55	0%	0%	0%	59%	100%	41%
CI = 95	0%	0%	0%	10%	100%	90%

layers are 600 and 1000 Mbps, respectively. CPU resources of the 1st-, the 2nd-, and the 3rd-layer servers are 6, 12, and 30 (GHz), respectively. Other setting parameters can be found in [10]. For the benchmark, other solutions, i.e., the energy-efficient offloading (EEO) [10] and non-offloading solutions are used to compare with the proposed ODO solution.

4.1 Scenario 1: When the System Load Is Low

We first evaluate the performance of the proposed ODO scheme under the setting that the system load is low, i.e., $\rho \in (0, 0.3)$. In Figs. 2(a), 2(b), and 2(c), we show the variation of the probabilities of guaranteeing service delay requirement corresponding with different levels of computational intensity (CI = (25, 55, 95)). In general, the probabilities of guaranteeing delay obtained by the proposed ODO, EEO, and non-offloading solutions increase as the service delay increases in the range of 5 ms to 30 ms.

Considering the computational intensity at medium and high levels, i.e., CI = 55 and CI = 95, as illustrated in Fig. 2(b) and Fig. 2(c), the non-offloading solution obtains the lower performance compared with the proposed ODO and EEO solutions. It is due to the fact that in the non-offloading solution, only the servers deployed at the 1st layer process the computational load, while the others perform offloading the computational load to the 2nd and 3rd layer servers. Thus, the non-offloading has the lower probabilities of guaranteeing the service delay requirement than the other solutions. By contrast, when CI = 25, the probability of guaranteeing service delay requirement of the EEO solution is lower than those of the other solutions. It is because the EEO solution tends to process the computational load at high layer servers to optimize the consumption energy. Meanwhile, when the CI = 25, the non-offloading solution offers a slightly higher probability of guaranteeing service delay requirement than the proposed solution as it does not suffer the transmissions delay among servers.

In Table 1, we demonstrate the percentage of the requests processed by servers at different levels of CI. It can be seen that the EEO solution prefers selecting the high-layer servers to handle the user requests rather than processing them at the lower-layer servers. It is because the high-layer servers have high computing energy efficiency. Thus, it can help this method optimized in terms of energy consumption. In particular, with the low computational intensity, i.e., CI = 25, 100% of user requests are processed at the 3rd layer servers to guarantee the energy requirement as shown in Table 1. However, this causes a high delay due to information exchange between layers. By contrast, the proposed ODO offloading solution processes 86% and 14% user requests at the 1st layer and 2nd layer servers, respectively. By this way, our proposed solution can obtain higher QoS in terms of service delay than the EEO solution. When the computational intensity increases, the 1st layer servers cannot handle user requests. Thus, in the proposed ODO solution, the computational tasks are offloaded to the 2nd and 3rd layer servers. Specifically, when CI = 55, 59% and 49% of requests are processed at the 2nd and 3rd layer servers, respectively. Similarly, when CI = 95, 90% and 10% of requests are handled at the 2nd and 3rd layer servers, respectively. Meanwhile, the EEO solution assigns 100% computational tasks for the 3rd layer servers. However, as the system load is small, i.e., $\rho = (0, 0.3)$, the queuing delay is small and insignificant when compared with the other delays. Therefore, the performance of both solutions is as the same at high CI levels when the required delay is greater than or equal to 15 ms.

4.2 Scenario 2: When the System Load Is Medium

We then investigate the performance of our proposed ODO solution when the system load is medium, i.e., $\rho \in (0.3, 0.6)$. Generally, the ODO solution always obtains higher probability of guaranteeing service delay requirement than the others. In particular, as CI = 25, the EEO solution achieves the lowest performance among three solutions. That is because it prefers handling user requests at high-layer servers. Thus, it suffers a higher transmissions delay than the others using low-layer servers to perform the computation tasks. How-

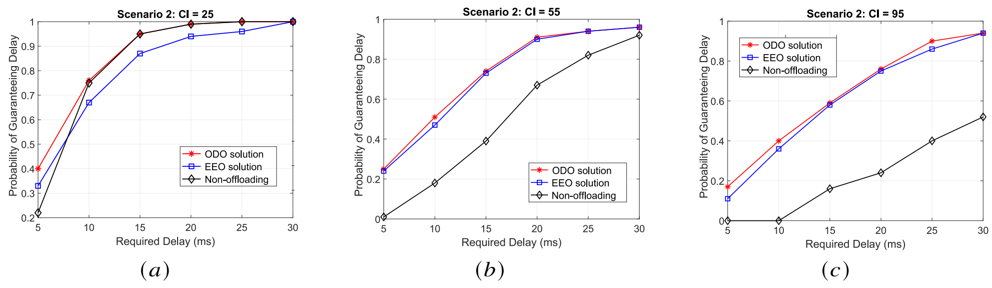

Fig. 3 Probability of guaranteeing service delay as (a) CI = 25, (b) CI = 55, and (c) CI = 95 with the medium system load.

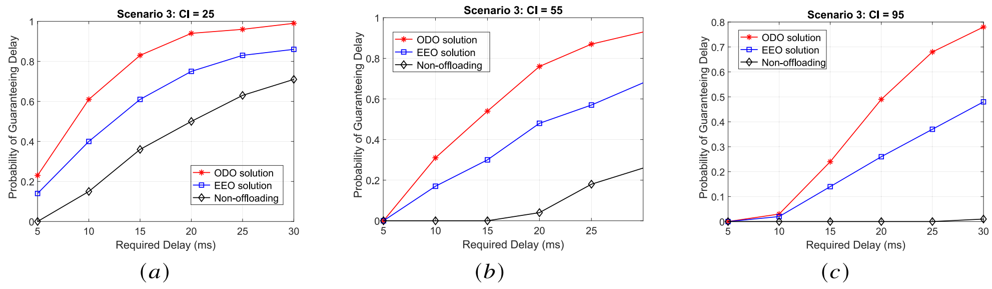

Fig. 4 Probability of guaranteeing service delay as (a) CI = 25, (b) CI = 55, and (c) CI = 95 in the case of high system load.

ever, when computation intensity increase, i.e., $\mathrm{CI} = \{55, 95\}$, the queuing delay is superior to the transmissions delay between servers and thus, the service delay increases. Therefore, the non-offloading solution shows a significantly lower performance compared with the others. That is because the non-offloading solution uses the 1st layer servers that have low computing capacity, to compute user requests. Thus, it suffers a higher queuing delay than the others.

4.3 Scenario 3: When the System Load Is High

Finally, we perform simulations to evaluate the performance of all solutions when the system load is high, i.e., $\rho = (0.6, 1)$ as shown in Fig. 4. In general, our proposed ODO solution continues showing the highest probability of guaranteeing service delay requirement among the others. That is because it has an optimized offloading strategy to share the computation tasks to servers under different levels of CI. Meanwhile, the EEO solution is no longer efficient in guaranteeing service delay requirement. It is because this method prefers using only servers at high layer that always overloaded. Thus, it leads to the high queuing delay.

5. Conclusion

In this paper, we have studied the offloading strategies in the hierarchical MEC networks. To improve the service delay in the MEC networks, we have proposed the server selection offloading scheme that optimizes both the queuing delay at servers and transmissions delay among servers under different levels of computational intensity. Numerical analyses have shown that the higher the system load and computational intensity, the more efficient the proposed optimal delay offloading method is compared with other baseline methods.

References

[1] A. Shakarami, M. Ghobaei-Arani, and A. Shahidinejad, "A survey on the computation offloading approaches in mobile edge computing: A machine learning-based perspective," Comp. Net., vol.182, 107496, 2020.

[2] Y.C. Hu, et al., "Mobile edge computinga key technology towards 5G," ETSI white paper, vol.11, no.11, pp.1–16, 2015.

[3] ETSI, "Mobile-Edge Computing–Introductory Technical White Paper," Available at: https://portal.etsi.org/Portals/0/TBpages/MEC/Docs/Mobileedge_Computing_-_Introductory_Technical_White_Paper_V1%2018-09-14.pdf

[4] J. Guo, Z. Song, Y. Cui, Z. Liu, and Y. Ji, "Energy-efficient resource allocation for multi-user mobile edge computing," IEEE GLOBECOM, Singapore, pp.1–7, 2017.

[5] S.J. Bigelow, "What is edge computing? Everything you need to know," Dec. 2021. Available at: https://www.techtarget.com/searchdatacenter/definition/edge-computing

[6] K. Zhang, Y. Mao, S. Leng, Q. Zhao, L. Li, X. Peng, L. Pan, S. Maharjan, and Y. Zhang, "Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks," IEEE Access, vol.4, pp.5896–5907, 2016.

[7] E. Meskar, T.D. Todd, D. Zhao, and G. Karakostas, "Energy-efficient offloading for competing users on a shared communication channel," IEEE ICC, pp.3192–3197, June 2015.

[8] C. You, K. Huang, H. Chae, and B.H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," IEEE Trans. Wireless Commun., vol.16, no.3, pp.1397–1411, March 2017.

[9] S. Wang, M. Zafer, and K.K. Leung, "Online placement of multi-component applications in edge computing environments," IEEE Access, vol.5, pp.2514–2533, 2017.

[10] S. Thananjeyan, C.A. Chan, E. Wong, and A. Nirmalathas, "Mobility-aware energy optimization in hosts selection for computation offloading in multiaccess edge computing," IEEE Open J. Commun. Soc., vol.1, pp.1056–1065, July 2020.