LUPINE PUBLISHERS

Current Trends in Computer Sciences & Applications

DOI: 10.32474/CTCSA.2022.02.000136

ISSN: 2643-6744

Research article

Design of Adaptive Offload Decision and Resource Allocation Algorithm for Critical and Non-critical Tasks

Yen Wen Chen* and BoHan Huang

Department of Communication Engineering, National Central University, Taiwan

***Corresponding author:** Yen Wen Chen, Department of Communication Engineering, National Central University, Taiwan

Received: March 29, 2022 **Published:** April 11, 2022

Abstract

The main concept of metaverse is to connect augmented reality (AR) and virtual reality (VR) devices to achieve common share space for new applications and services. The existing internet of things (Iot) services shall support high interaction system architecture and techniques to respond to this application trend. Although the cloud computing based network services can support high and flexible computing and storage space, it still cannot satisfy the above real time interaction demand due to long transmission delay. The edge computing architecture is therefore proposed for task offloading requested by the clients to effectively utilize the computing resource provided by the edge server and save the energy of the client device. However, the communication and computing resource are limited and shall be properly arranged for tasks with different demands. In this paper, we propose the Load-Adaptive and Joint Resource Allocation (LAJRA) offload decision algorithm by considering the integrated arrangement of computing and communication resource for the offloading requests from Iot devices. Exhaustive simulations were performed to examine the performance of the proposed algorithm. The simulations results show that the proposed algorithm can effectively utilize the computing resource and provide much higher acceptance rate for the critical tasks.

Introduction

The rapid and mass deployment of internet of things (IoT) devices enables the human activities to be further integrated with the cyber space. The metaverse services can then be realized by applying the information provided by various IoT devices in the virtual shared space. The metaverse services always need real time interaction/response. The response time of the service provisioning mainly relies on time required for data processing and communication. Moreover, computing resource of IoT devices is limited and their energy is constrained. Therefore, the task offloading is required for several IoT oriented services. The cloud server provides huge and flexible computing and storage resource, which is functionally meaningful for the task offloading from the end device point of view [1]. However, the performance, especially the delay time, is the most concerned issue for real time services. The delay time includes the processing delay and the transmission delay. And the cloud server, in addition to providing computing and storage resource for its client, being a centralized platform, it needs some overhead to support the scheduling and management of the shared resource for the demands from huge number of devices. Additionally, the transmission path between the client or user equipment (UE) and cloud server needs to travel through the access network, core network, and public internet, the long latency is not acceptable by some real time services. Thus, the main contribution of cloud server is to provide flexible and reliable computing and storage resources for UEs, however, it is hard to provide real time services.

In order to provide increasing real time IoT services, the edge-computing concept is proposed to compensate the existing cloud computing architecture [2,3]. The edge computing extends the traditional cloud capabilities at the edge of the network to perform delay sensitive tasks. Thus, the edge server can be deployed beside the base station to provide the offloading service for the UE within the coverage area. Comparing to the cloud computing, the edge computing serves much less clients and the transmission latency is much more reduced. The edge server can provide efficient offload service for the local UEs through distributed manner. In addition to the time critical task, the IoT device may offload the task due to its limited energy. Therefore, the computing resource of the edge server shall also be well allocated and scheduled to meet the demand of the time sensitive tasks because the computing power of edge server is limited when comparing to the cloud server. The procedure of offloading can be divided into the transmission part

and processing part. The time required to complete the offloaded task includes the transmission delay and processing delay. Moreover, the uplink resource of the wireless network is contention based, which may cause uncertain delay. Therefore, the network side needs to jointly take the computing resource and transmission resource into consideration to see whether the overall delay can satisfy the required time constraint or not. In addition, this is the main objective of the proposed algorithm in this paper. This paper organized as follows. In Section II, we provide the overview of related studies. The proposed LAJRA offload decision algorithm is described in Section III. Moreover, the experimental simulation results are shown in Section IV with discussion. Finally, the conclusions and future works are summarized in the last section.

Overview Of Related Works

The main objective of cloud computing architecture is to provide resources sharing so that user can request for computing and storage for temporal allocation and pay for the resource actually used. Several "x as a service" (xaaS) concepts were proposed to achieve various demands from users [4]. Although cloud computing can provide flexible resource for user to offload tasks, it is hard to meet the task with real time applications. Comparing to the cloud computing, the computing resource of edge server is much less, than cloud server, therefore, the issue resource allocation for critical tasks is the main study issue in edge computing environment. The hybrid cloud computing and edge computing architecture, as shown in (Figure 1) is more suitable for practical applications [5,6].

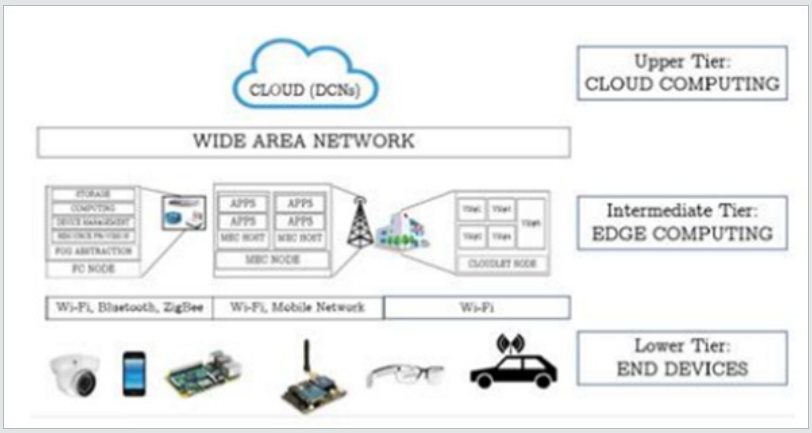

Figure 1: The hybrid cloud/edge computing architecture [5].

In order to provide offloading for tasks with either time constraint or power constraint, the edge server shall support proper admission policy and effective resource allocation for the issued offload tasks with different needs [6]. In [7], the authors formulated the optimization of the computation offloading for an edge computing system and reviewed the collaborative scheduling methods. In [8], the existing works on resource optimization in federated learning were reviewed, and an elastic learning framework for emerging applications in edge computing and the Internet of Things was proposed. The cost- efficient in-home health monitoring system for Internet of Medical Things (IoMT) was proposed in [9]. The authors proposed the decentralized non-cooperative game approach to minimize the system-wide cost and prove that the proposed algorithm can reach a Nash equilibrium. Although the above studies proposed several effective resource allocation schemes for the edge server from either optimization or machine learning approaches, the offloading requests always arrive randomly in sequence not in batch in practical environment. The resource allocation is hard to be optimized due to unpredictable requests. Further, in addition to the computing resource of the edge server, the UE needs to contend the uplink radio resource for task offloading and, therefore, the wireless resource allocation shall be properly considered as well. In addition, this is the main study issue of the proposed Load-Adaptive and Joint Resource Allocation (LAJRA) algorithm.

The Proposed LAJRA Algorithm

We assume that the UE may request for offloading to save its computing power and energy. Moreover, the offloaded tasks can be further divided into two types, one is the delay-required task (DT), and the other is the non-delay-required task (NDT) in the wireless environment with one base station as shown in (Figure 2). The critical task is defined as the task that has delay constraint and cannot be completed in the local UE side due to either insufficient computing power or limited energy. The server shall respond whether to accept the task or not after receiving offload request. According to the delay requirement, local CPU capacity, and remaining energy, we classify the offload tasks into five types as shown in (Figure 3). Among them, type 2, 3 and 5 are identified as the critical tasks because they cannot complete their work without the assistance of edge server due to either the lack of local computing power or the remaining energy. Moreover, type 2 is NDT while types 3 and 5 are DT. DT tasks because they are DT and cannot be completed under the delay constraint as mentioned

in previous section, the requesting task arrives randomly and it is hard to predict the task type of the upcoming request. If the resource, either the computing power or the wireless bandwidth, is congested and cannot be allocated to the arrival request, then the offload request will be rejected if the task is non critical. However, fir the critical task, if the offload task is DT, then the proposed algorithm will seek for the possibility to preempt the resource that has been allocated for the NDT task. It is noted that the proposed algorithm shall respond the UE whether the task is accepted or not within a predefined acceptable delay when receiving the request offloading. This delay period is designed to confirm those NDT tasks, which have been allocated with bandwidth, shall be accepted for offloading and will not be preempted. The use of this delay period in the proposed algorithm will be discussed later. As the allocated resource of the accepted NDT task cannot always be preempted by the DT task, we propose two steps toward the offloading procedure. As shown in (Figure 4), the first step is to decide whether the arrival task is accepted for the following resource allocation or not, and the step 2 performs the resource allocation.

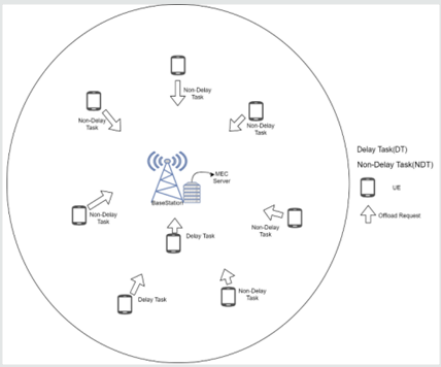

Figure 2: System model.

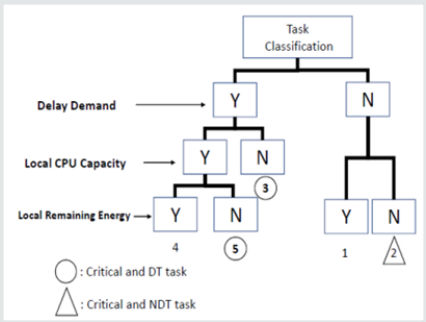

Figure 3: Task classification.

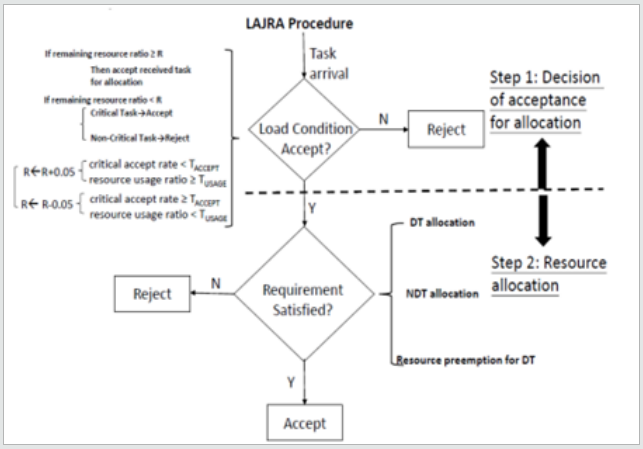

Figure 4: Procedure of the LAJRA algorithm.

In step 1, the non-critical task is not allowed to enter step 2 if the remaining resource ratio (either computing resource or the wireless bandwidth) is less than a threshold R. In addition, the threshold is adaptively changed according to the accept rate of critical tasks and the resource usage ratio during the collected statistic of the latest period. If the acceptance rate is smaller than TACCEPT and the resource usage ratio is less than TUSAGE, the value of R will be increased by 0.05 till 1, and it will be decreased if the accept rate is higher and the resource usage ratio is smaller. The resource allocations for DT and NDT are not the same. The DT task is allowed to conditionally preempt the resource from NDT task in Step 2.

The resource considered in this paper include the computing resource provided by the edge server and the bandwidth of the wireless network. Therefore, the proposed algorithm shall examine both of the computing power and network bandwidth to see whether the residuals of both resources are sufficient to support the requesting task or not. Here we ignore the downlink resource allocation because that the processed result is always much smaller than the uplink data volume and the downlink resource can be effectively managed by the base station. The summation of the waiting times, including the time before transmission and the time before processing when it received by the edge server, the transmission time, and the processing time shall be less than the tolerable delay time of DT task. Additionally, the allocation for uplink time shall be prior to the start of the processing time. Here we propose two allocation alternatives for DT task in the proposed LAJRA algorithm as shown in (Figure 5). The nearest first allocation allocates the available resource as closer to the current time as possible; however, the just enough allocation approach starts to allocate the available resource from far-end, by referring to its tolerable delay time, to the near-end.

In the example of (Figure 5), the requesting task arrives at 30ms and its delay requirement is 200ms. Then the nearest first allocation scheme checks and allocates the available resource from 30ms up to 230ms; while the just enough scheme arranges the available resources from 230ms down to 30ms. For the allocation of NDT task, although there is no delay time constraint for NDT task, it is not possible to postpone the execution of the NDT task after a long time later. The parameter is defined as the allocation interval for NDT tasks. Figure 6 shows the cases of acceptance and rejection of NDT task due to enough resource and lack of resource, respectively. In the proposed LAJRA, the DT task can preempt the resource that has been allocated to NDT; however, there is a constraint. As mentioned above, the edge server shall respond the acceptance or rejection of the requesting task. Hence, we define the lock time RLOCK for the non-preemption interval. Thus, the UE that has the allocated resource within this interval shall have been responded with the acceptance of its offloading request and all resource allocated in this interval shall not be preempted. Figure 7 illustrates the examples for the situations that allows preempting and denies preempting. (Figure 7) shows that a DT offload requests are received by the system at 40ms and 240ms, respectively, and the RLOCK is assumed 100ms. The upside of (Figure 7) shows that the DT can preempt the resource that has been allocated for the NDT and can be accepted by the system, and the NDT task will be rejected. The downside of (Figure 7) illustrates that the DT task cannot preempt the resource of NDT that has been allocated within the lock interval.

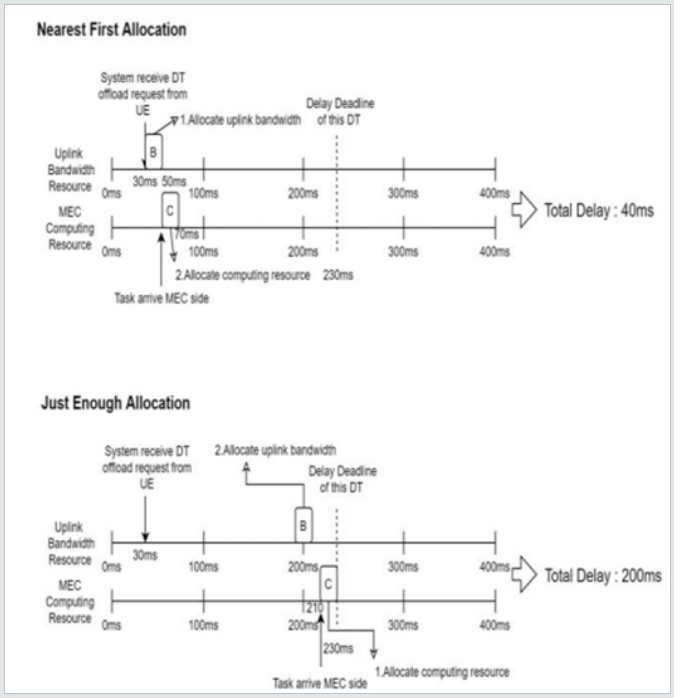

Figure 5: Resource allocation for DT task.

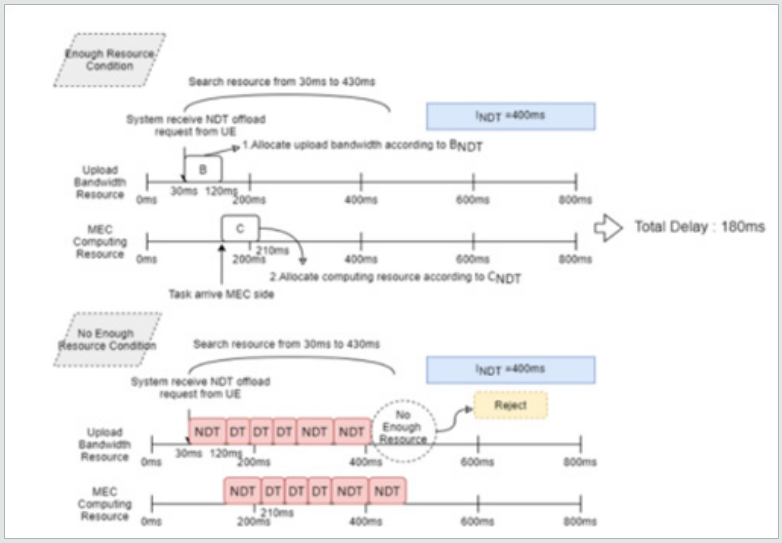

Figure 6: Resource allocation for NDT task.

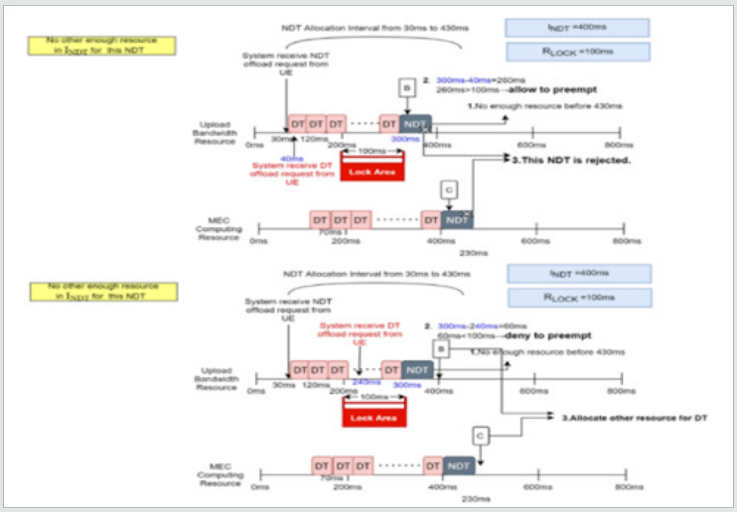

Figure 7: Examples of resource preemption in LAJRA.

Experimental Simulations

In order to examine the effectiveness of the proposed algorithm, exhaustive simulations were performed. The common parameters used during the simulations are provided in Table 1. Figure 8(a) and Figure 8(b) illustrate the average delay of nearest first allocation and just enough allocation for DT and NDT, respectively. Here we let the parameters of INDT, RLOCK be 600ms, 0ms and the number of DT tasks is 20% of all generated tasks during the simulations The simulation result of (Figure 8a) shows that although the DT task of the just enough approach has higher average delay than that of the near first approach, it can still meet the desired delay requirement. And the average delay of NDT tasks of the just enough approach is a little less than that of the nearest first allocation scheme because the NDT task tends to allocate the nearer RB, however, the difference is not significant. The simulation results of the proposed approaches is compared to that of the critical-only scheme. The critical-only scheme rejects all the requests of non-critical tasks and allocates the resource to the critical tasks only. Figure 9 shows the comparison results. The results illustrate that the critical-only scheme has higher accept rate than the proposed algorithm for the critical tasks in (Figure 9a) because the critical-only approach will not accept the offload request of non-critical tasks and all resources will be exclusively allocated for DT tasks. Although he proposed algorithm could accept only 90% of the critical tasks, however, it accepts much more non-critical tasks as shown in (Figure 9b). The accept rate of critical task can be further improved by adjusting the thresholds of TACCEPT and TUSAGE. (Figure 10) provides the detail accept rates of each request type. It shows that the critical tasks has much higher accept rate than that of non-critical tasks. As the simulation results shown in (Figure 9), the critical-only allocation scheme can accept above 98% of the critical tasks when the number of UE is less than 50, however, its resource utilization is less than that of the proposed scheme as shown in (Figure 11). It is noted that the resource is not fully utilized is mainly due to the fragment of resource during allocation.

Table 1: Simulation parameters.

Parameters	Descriptions
Inter-Arrival Time of Task	Exponential(30ms)
Date Size of Task	U(0.8Mb,1.2Mb)
Computing Size of Task	(0.5 * Data Size of Task) GHz
Delay Demand of DT(DDT)	U(150ms,250ms)
Total Upload Bandwidth of eNB	100Mbps
Total Computing Capacity of MEC	50GHz/s
Local Computing Capacity of UE	1.5GHz/s
The Bandwidth of Allocating NDT(BNDT)	50Mbps
The Computing Capacity of Allocating NDT(CNDT)	25GHz/s
The acceptable rate threshold of critical task (TACCEPT)	90%
The usage rate threshold of bandwidth (TUSAGE)	75%

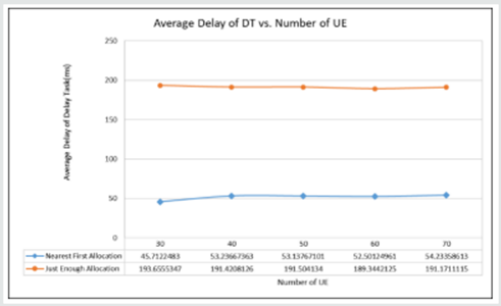

Figure 8: The comparisons of nearest first allocation and just enough allocation approaches.

Figure 8a: Average delay of DT tasks.

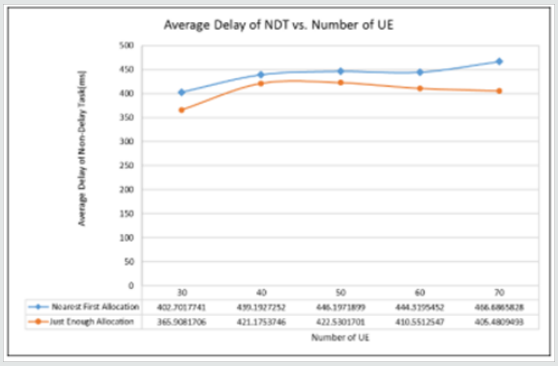

Figure 8b: Average delay of NDT tasks.

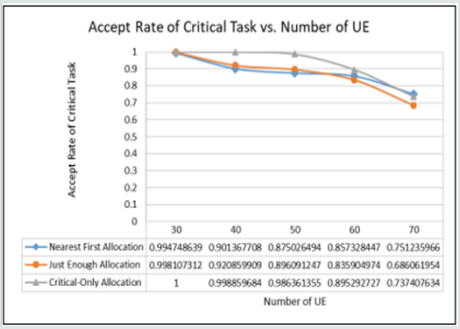

Figure 9: Comparisons of the proposed algorithm and the critical-only allocation scheme.

Figure 9a: Accept rate of critical tasks.

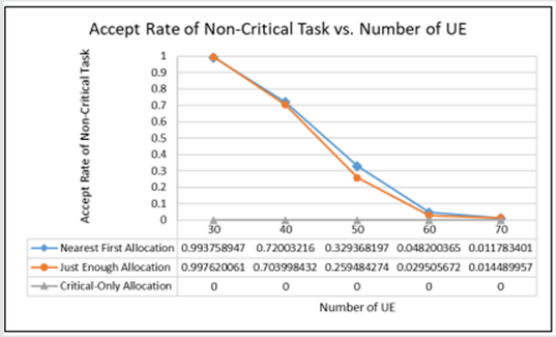

Figure 9b: Accept rate of non-critical tasks.

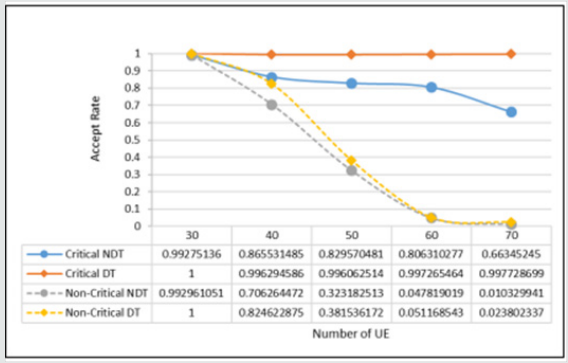

Figure 10: Accept rates of each request types.

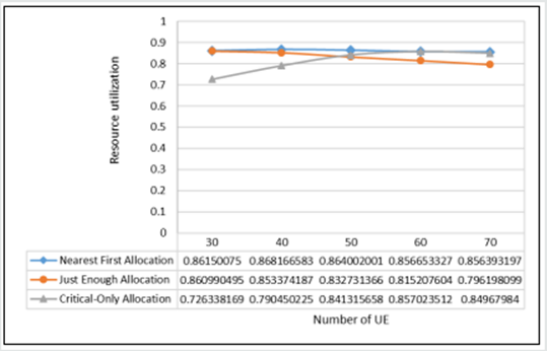

Figure 11: Comparison of resource utilization.

Conclusions

In this paper, the LAJRA algorithm is proposed to allocate the computing and bandwidth resource for the acceptance decision of task offload in edge computing environment. The requesting tasks are classified into critical tasks and non-critical tasks. The proposed algorithm flexibly adjusts the threshold to filter the tasks to get into allocation by considering the accept rate of critical tasks and the resource utilization. Moreover, the critical task can conditionally preempt the resource that has already been allocated to the NDT tasks by properly define the acceptance/rejection responding time RLOCK. The simulation results indicate that the proposed algorithm can properly tradeoff the resource allocation between critical and non-critical tasks and achieves high resource utilization. However, this paper only considers the single edge server scenario. It can be extended to study the offloading in mobile environment with multiple base stations and edge servers. As UE may travel around base stations, the issue of effective offloading decision and resource allocation is more complex and it is one of our ongoing study issues.

References

1. D Puthal, BPS Sahoo, S Mishra, S Swain (2015) Cloud Computing Features, Issues, and Challenges: A Big Picture. International Conference on Computational Intelligence and Networks pp. 116- 123).
2. N Abbas, Y Zhang, A Taherkordi, T Skeie (2018) Mobile Edge Computing: A Survey, in IEEE Internet of Things Journal 5(1): 450-465.

3. N Hassan, KA Yau, C Wu (2019) Edge Computing in 5G: A Review, in IEEE Access. 7: 127276-127289.

4. Microsoft, Describe different cloud services.

5. K Dolui, SK Datta (2017) Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing, in Global Internet of Things Summit (GIoTS).

6. K Cao, Y Liu, G Meng, Q Sun (2020) An Overview on Edge Computing Research, in IEEE Access 8: 85714-85728.

7. Chen, Shichao, Qijie Li, Mengchu Zhou, Abdullah Abusorrah (2021) Recent Advances in Collaborative Scheduling of Computing Tasks in an Edge Computing Paradigm. Sensors 21(3): 779.

8. R Yu, P Li (2021) Toward Resource-Efficient Federated Learning in Mobile Edge Computing. IEEE Network 35(1): 148-155.

9. Z Ning (2021) Mobile Edge Computing Enabled 5G Health Monitoring for Internet of Medical Things: A Decentralized Game Theoretic Approach. IEEE Journal on Selected Areas in Communications 39(2): 463-478.

DOI: 10.32474/CTCSA.2022.02.000136