QoS-aware Mobile Edge Computing System: Multi-server Multi-user Scenario

Te-Yi Kan, Yao Chiang, Hung-Yu Wei
Dept. of Electrical Engineering
National Taiwan University
Taipei, Taiwan
hywei@ntu.edu.tw

Abstract—**Mobile/Multi-access Edge Computing (MEC) system is a kind of fog computing system to provide computation resources at the edge of Radio Access Network (RAN). In this paper, we discuss offloading decision and resource allocation of both radio resources and computation resources which are both the crucial issues in MEC system. Moreover, we study a multi-server system which is more consistent with practical MEC system in the future rather than single-server system, and load distribution is introduced into our system to leverage the computation resources of MEC servers effectively. The objective of our work is to promote QoS, and thus we formulate a cost minimization problem by modeling QoS of the end users with our self-defined cost function. A heuristic algorithm is developed and compared with other schemes. Numerical results verify that our proposed algorithm significantly enhances QoS and gains a large performance gap over other schemes.**

Keywords—fog computing, mobile/multi-access edge computing, task offloading, resource allocation, load distribution

I. Introduction

Nowadays, the evolution of innovative mobile applications, such as augmented reality and interactive game, goes hand-in-hand with the evolution of mobile devices. Those applications attract great attention because of their abilities to bring convenience to and spice up our lives. However, mobile devices are typically resource-constrained. They don't have enough resources to run those applications. To this end, task offloading is widely used to ease off the stress of mobile devices in energy consumption and computation resource consumption. Multi-access/Mobile Edge Computing (MEC) system [1], a kind of fog computing system proposed by ETSI (European Telecommunications Standards Institute), is a brand-new and promising paradigm to offer an environment characterized by low latency as well as provide computation resources for mobile devices to liberate them from intensive computation applications. The traditional Mobile Cloud Computing (MCC) system can also provide computation resources, but running applications on the cloud server may incur large latency due to the long distance from mobile devices to the cloud server. With the characteristic of low latency, MEC is suitable for real-time applications.

There are three critical issues, offloading decision, resource allocation, and load distribution, on the incorporation of task offloading into MEC system. First, to make our MEC system effective, a decision on task offloading to determine whether the offloading is profitable in terms of energy consumption or execution delay is an important topic. For instance, [2] incorporated dynamic voltage scaling (DVS) technology into computation offloading. The authors investigated latency minimization problem (LM) and energy consumption minimization problem (ECM) in a single server scenario. By jointly optimizing the computation speed, transmit power and offloading ratio of mobile devices, they finally proposed the LPCO algorithm for LM and the EPCO algorithm for ECM. Furthermore, the original single server system was extended to the multi-server system. To deal with low-latency requirement of novel mobile applications, the incorporation of Cloud radio access network (C-RAN) and Multi-access/mobile edge computing (MEC) was discussed in [3]. With MEC system, the new paradigm was more suitable for the low-latency application. In this paper, dozens of experiments and discussions were done. According to these discussions, the authors concluded that generalized-processor sharing (GPS) model is more appropriate to practical General Purpose Processor (GPP) Platform than other models. Moreover, Two-Thresholds Forwarding Policy (TTFP), which considered CPU utilization and traffic intensity, was proposed to handle the data traffic between the cloud center and MEC system. Several papers discussed the tradeoff between energy consumption and latency, such as [4] and [5]. In [4], an in-depth theoretical analysis on the tradeoff between latency and the energy consumption was offered. The authors studied a single UE femto cloud system and investigated partial offloading decision problem with MIMO technology. With joint optimization of the radio resources and computation resources, the optimal communication strategy, as well as the optimal load distribution between mobile devices and femto access points (FAPs) was obtained. The power-delay tradeoff was also investigated in [5]. The authors studied a multi-user mobile edge computing system and formulated a power consumption minimization problem with task buffer stability constraints. By optimizing CPU-cycle frequencies of local devices, transmit power and bandwidth allocation, an online algorithm was proposed.

In addition, the radio resources for connecting to the MEC server are deficient since dozens of devices desire to connect with the wireless network. Meanwhile, compared with the cloud server, computation resources on the MEC server are relatively limited. Therefore, proper mechanisms for resource allocation of radio resources and of computation resources are essential. For example, reference [6] investigated the radio resource allocation problem. The authors jointly optimized offloading decision and radio resource allocation in order to minimize the computation cost. The heuristic algorithm was proposed by adopting game theory. With this algorithm, each mobile device can make its decision in the distributed fashion. On the other hand, [7] worked on computation resource allocation. The authors designed a utility function related to energy consumption and latency, and formulated a utility maximization problem. Then, they introduced Lagrange multiplier to distribute computation resource to each offloaded task and address the utility maximization problem. Moreover, resource allocation of both two types of resources are discussed in [8] and [9]. The objective of [8] is to minimize the completion time by optimizing radio and computation resources. [8] studied two access schemes, FDMA and TDMA, and proposed corresponding algorithms. [9] aims at minimizing energy consumption to broaden the battery life of mobile devices. Same as [8], two access schemes, OFDMA and TDMA, were considered in [9]. Two heuristic algorithms, offloading priority function and average offloading priority function, were developed for TDMA and OFDMA respectively.

Last but not least, multi-server scenario also attracts great attention. With the interaction between MEC servers, MEC servers can share their computation resources and distribute some workload from the servers with inadequate resources to other servers. With load distribution, the performance of MEC system can be improved as well as the QoS of the end users can be promoted. [10] studied a multi-user, multi-cell, multi-cloud system and investigated load distribution in a predefined cluster of clouds. The objective is to minimize the overall energy consumption at mobile terminal side as well as meet the latency constraints at the same time. The authors formulated a joint optimization problem of the radio and computation resources. Nonetheless, this problem is non-convex in both the objective function and the constraints, so they proposed a heuristic algorithm with convexification of both the objective function and the constraints. Reference [11] considered both load distribution and clustering problem. The cluster for each task is dynamically built in order to reduce the energy consumption of the MEC servers. In addition, different objectives were discussed in [11] and the author developed different greedy algorithms in terms of different purpose. The main structure of these algorithms can be summarized as two steps. First, the cluster for each task will be dynamically constructed. Second, the task will be distributed to the proper server in its cluster according to different purposes of algorithms.

In this paper, we study the above three critical problems at the same time, which is seldom discussed in related works. We optimize full offloading decision in terms of QoS of end users. Instead of optimizing average delay [12], we take the residual energy of mobile devices and the variety of delay tolerance of tasks into account by designing a cost function to model QoS. With this cost function, the original delay minimization problem becomes a cost minimization problem. Resource allocation of both radio and computation resources is also studied in our work. There have been a few papers optimize both radio resources and computation resources, such as [8] and [9]. However, instead of partial offloading studied in the related works, we investigate full offloading from mobile devices to the MEC server. We have investigated this problem in our previous work [13]. But, when the number of devices scales up, the computation resources of the single MEC server will become insufficient so that some tasks will be terminated. Therefore, the QoS of users will decrease sharply as the devices set becomes larger. To this end, we introduce the multi-server scenario and incorporate load distribution into our MEC system to enhance the QoS in our new work. Especially, we consider both full task distribution and partial task distribution in our load distribution scheme. On the other hand, our algorithm can not only enhance QoS but also reduce the energy consumption on MEC servers and the backhaul flow. The reason is that our method intends to execute tasks on the original server rather than distributing them to other servers. By this mechanism, the tasks will be offloaded to other servers only when the resources of the original server are deficient and hence we can save the capacity of backhaul and prevent it from wasting the energy of other servers. The performances of our algorithm are evaluated and compared with other schemes. The results show that our algorithm reduces more cost and obtain better performance in comparison with other algorithms.

The rest of this paper is organized as follows. The system model and our heuristic algorithm are presented in Section II and Section III respectively. Then, we evaluate our proposed algorithm and show our simulation results in Section IV. Finally, we conclude this paper in Section V.

II. SYSTEM MODEL

In this paper, we study a multi-server multiuser Mobile/Multi-access Edge Computing system as shown in Fig. 1. In our system, we have a set of base stations (BSs) $\mathcal{M} = \{1,2,\dots,M\}$ which consists of M BSs, and each BS s is equipped with a MEC server with computation capacity F_s. In addition, each BS s serves N_s mobile device and these mobile devices can be described as $\mathcal{D}_s = \{D_1^s, D_2^s, \dots, D_{N_s}^s\}$. Each MEC server is connected with each other with Mp3 interface and there is one MEC controller to manage all the MEC servers with Mm5 interface. Here, Mp3 and Mm5 are the interfaces defined by ETSI in its MEC related document. This MEC controller is responsible for allocating computation resources inside each MEC server and distributing workload among all the MEC servers. For each mobile device $n \in \mathcal{D} = \bigcup_{s=1}^{M} \mathcal{D}_s$, it has one task $L_n = \{a_n, d_n, T_n^{max}\}$, where a_n denote the input data size of L_n, d_n is the number of required CPU cycles of L_n, and T_n^{max} represents the delay tolerance of L_n. We assume that the output data size of a task is extremely smaller than the input data size, so we omit the cost of sending back the result. Moreover, each mobile device has its own residual energy E_n^R and computation capacity f_n. In our work, each task can select either local execution or offloading. Noted that we consider full offloading between mobile devices and MEC servers. Hence, each task has its offloading decision $x_n \in \{0,1\}$. $x_n = 0$ represents the task selects to be executed on the local device. Otherwise, $x_n = 1$ represents the task will be offloaded to the MEC server. In the following two subsections, we will present our schemes of local execution and remote execution (offloading). Then, we will formulate our optimization problem.

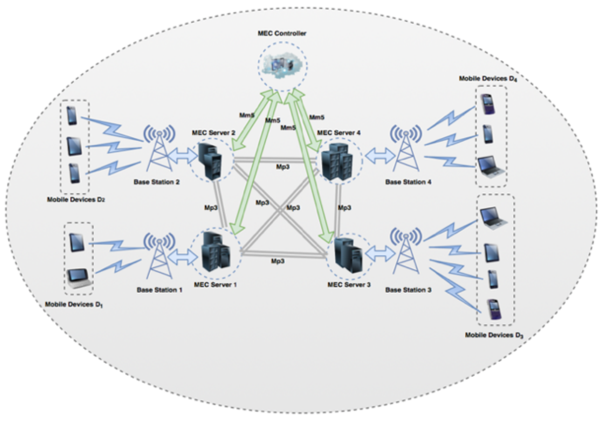

Fig. 1. System model

A. Local execution

For local execution, we adopt the same scheme in our previous work [13]. We talk through our scheme and the formulation in Appendix.

B. Remote execution

If the task selects offloading, it will follow the remote execution scheme. In remote execution, there are two major steps, wireless communication and task execution. Each device which desires to offload its task will first acquire one channel to transmit data to the MEC server. In this step, we also follow the same mechanism in [13] as shown in Appendix. Then, each offloaded task should obtain some computation resources on the MEC servers to execute. Since we consider multi-server scenario and investigate load distribution, each offloaded task has its own server decision $e_n \in \mathcal{M}$. Furthermore, we assume that our MEC controller can further divide a task into some instructions. Therefore, a task can not only be fully distributed to another server but also be partially offloaded to other servers. We define the MEC server which serves the task L_n is Local MEC Server and the other servers are called Nearby MEC Server.

1) Full task distribution: The task in this scheme will be executed on either Local MEC Server or Nearby MEC Server. Especially, when the task L_n is offloaded to Nearby MEC Server, there will be an additional transmission time a_n/R, where R is the transmission rate between two MEC servers. Nevertheless, we assume R is a great value, so this transmission time can be ignored. According to this assumption, the execution time on either Local MEC Server or Nearby MEC Server can be obtained as

$$T_n^{exe} = d_n / y_n^{e_n} \tag{1}$$

Where $y_n^{e_n}$ is the computation resources allocated to the task L_n on the server e_n which L_n selects.

2) Partial task distribution: In this scheme, MEC controller will divide the task L_n into W_n instructions $L_{n,w} \in \{L_{n,1}, L_{n,2}, \dots L_{n,W_n}\}$. We can describe each instruction as $L_{n,w} = (a_{n,w}, d_{n,w}, e_{n,w})$ where $a_{n,w}$ is the input data size of the instruction, $d_{n,w}$ is the number of required CPU cycles of the instruction, and $e_{n,w}$ denotes the server decision of the instruction. In addition, there will be an extra instruction $L_{n,o}$ to gather the output results of the other instructions. Hence, this instruction $L_{n,o}$ can be executed only after all the other instructions have already been done. We describe this instruction as $L_{n,o} = (y_{n,0}, T_{n,o})$, where $y_{n,o}$ is the required computation resources of $L_{n,o}$ and $T_{n,o}$ is the execution time of $L_{n,o}$. Noted that we ignore the transmission time among the MEC servers. The execution time can be written as

$$T_n^{exe} = \max_{w \in \{1,..,W_n\}} \left(d_{n,w} / y_{n,w}^{e_{n,w}} \right) + T_{n,o} \tag{2}$$

Where $y_n^{e_{n,w}}$ is the computation resources allocated to $L_{n,w}$ on the server $e_{n,w}$ which $L_{n,w}$ selects. Especially, each $L_{n,w} \in \{L_{n,1}, \dots, L_{n,W_n}\}$ can be executed independently and simultaneously, so the execution time is the maximal complete time of those $\mathrm{W_n}$ instructions plus the complete time of $\mathrm{L}_{n,o}$.

C. Problem formulation

The main objective of our work is to minimize the completion time of tasks. In addition to our main objective, there are some features of tasks and mobile devices we take into account. First, each task has its own delay tolerance. For the tasks with lower delay tolerance, they should be executed in the shorter period of time. Second, the execution of the task cannot run out of the residual energy of the mobile device. In order to incorporate the above two features into our completion time minimization problem, we define a cost function as

$$\mathcal{C}_n = \begin{cases} \beta_n T_n; if\ T_n \le T_n^{max}\ and\ E_n < E_n^R \\ \varphi;\ otherwise \end{cases} \tag{3}$$

Where T_n is the completion time of the task L_n and E_n is the energy consumption of the task L_n. If L_n choose local execution ($x_n = 0$), $T_n = T_n^l$ and $E_n = E_n^l$. Otherwise, $T_n = T_n^{tx} + T_n^{exe}$ and $E_n = E_n^{tx}$. β_n is the weight to introduce delay tolerance to our optimization function and it is shown as

$$\beta_n = \frac{1/T_n^{max}}{\sum_i 1/T_i^{max}} \tag{4}$$

By this weight, minimizing cost will lead to much smaller completion time for those tasks with lower delay tolerance. Besides, we have two special cases. One is that the task completion time is longer than its delay tolerance. The other is that the execution of the task runs out of the residual energy of the mobile device. In these two special cases, the task will fail and incur constant cost φ. It is a remarkable fact that task execution failure is the worst result for end users. Thus, this cost is much larger than normal cost $\beta_n T_n$ in our settings.

According to the above cost function (3), we formulate the original completion time minimization problem as a cost minimization problem written as

$$\begin{aligned} &\min_{\boldsymbol{x},\boldsymbol{h},\boldsymbol{e},\boldsymbol{y}} Cost = \sum_{n \in \mathcal{D}} \mathcal{C}_n \\ s.t\quad & C_1: x_n \in \{0,1\}\ \forall n \in \mathcal{D} \\ & C_2: h_n \in \mathcal{H}_s \cup \{0\} \\ & C_3: e_n \in \mathcal{M}\ \forall n \in \mathcal{F} \\ & C_4: y_n^s \ge 0\ \forall n \in \mathcal{F} \\ & C_5: e_{n,w} \in \mathcal{M}\ \forall n \in \mathcal{P}, \forall w \in \{1,2,..,W_n\} \cup \{o\} \\ & C_6: y_{n,w}^s \ge 0\ \forall n \in \mathcal{P}, \forall w \in \{1,2,..,W_n\} \cup \{o\} \\ & C_7: \left(\textstyle\sum_{n \in \mathcal{F}} y_n^s + \sum_{n \in \mathcal{P}} \sum_w y_{n,w}^s \right) \le F_s\ \forall s \in \mathcal{M} \end{aligned} \tag{5}$$

Where $\boldsymbol{x}$, $\boldsymbol{h}$, $\boldsymbol{e}$, and $\boldsymbol{y}$ represents the vector of offloading decision, channel selection, server decision and allocated computation resources respectively. We have seven constraints C_1 to C_7. Constraint C_1 represents that each device can only select either local execution or offloading. Constraint C_2 states that each device should select one channel to offload its task or do not select a channel (local execution, $h_n = 0$). Constraint C_3 and C_5 ensure that each offloaded task or its instructions will be executed on one MEC servers $s \in \mathcal{M}$. Noted that $\mathcal{F}$ is the set of tasks which are offloaded and fully executed on one server and $\mathcal{P}$ is the set of tasks which are offloaded but further divided into small instructions. Constraint C_4 and C_6 guarantee that the allocated computation resources for each task or instruction will be a positive value or zero. Constraint C_7 states that the computation resources distributed from each server s should not exceed its computation capacity F_s.

III. PROPOSED ALGORITHM

In this section, we develop an algorithm to address the above cost minimization problem by optimizing offloading decision, resource allocation and load distribution. Our heuristic algorithm can be briefly divided into two parts. The first part will be performed on each MEC server independently. Otherwise, all the MEC servers will collaborate on load distribution in the second part. It's worth noting that MEC controller gathers the information and is in charge of making decisions of load distribution in the centralized aspect only in the second part.

Algorithm 1 Our Proposed Heuristic Algorithm

```
1: for Each MEC server $s \in \mathcal{M}$ do
2:     Classified devices set $D_s$ and assign priority $\phi_n$;
3:     Allocate channels to device in $S_{mec}$ in ascending order
       of their priority;
4: end for
5: for Each MEC server $s \in \mathcal{M}$ do
6:     Allocate resources $\Delta_n$ to tasks of devices belong to $S_{mec}$
       in ascending order of their delay tolerance until all the
       tasks are served or the residual resources are not
       enougth to serve more tasks;
7: end for
8: Assign the unserved task $i$ to the server with mimimal resid-
   ual resources $F_s^R > \Delta_i$ in ascending order of $\Delta_n$ until all the
   unserved tasks are served or no server has enough residual
   resources to serve tasks;
9: Consider partial task distribution for the tasks which are
   still unserved after performing full task distribution;
10: for Each unserved task in ascending order of $\Delta_n$ do
11:     Divide $L_i$ into small instructions $L_{i,w}$;
12:     Assign $L_{i,w}$ with lowest $\Delta_{i,w}$ to the server with minimal
        $F_s^R > \Delta_{i,w}$ until all instructions are served or no server
        can serve the following instructions;
13:     if $L_i$ has some instructions $L_{i,w}$ remain unserved then
14:         Terminate the execution of $L_i$;
15:     end if
16: end for
17: for Each MEC server $s \in \mathcal{M}$ do
18:     For each device belongs to $S_{other}$, decide wheter
        to offload its task or not;
19:     Assign channels to the devices desire to offload;
20:     Allocate resources to all the tasks assigned to this server;
21: end for
```

1) Inside the MEC server (First part): First, we will classify each device set $\mathcal{D}_s$ into S_{mec} and S_{other}. Then, we will assign priority value $\emptyset_n$ to each device according to our self-defined priority function. The mechanisms for device classification and priority determination are similar to the mechanisms in our previous work [13]. The detail can be found in Appendix and [13]. After that, devices in S_{mec} will obtain the channels in ascending order of their priority and acquire computation resources on the MEC server. For computation resource allocation, it required the cooperation of all the MEC servers and will be present in the second part. In the end, we will consider the offloading decision and resource allocation for the devices in S_{other} by following the similar mechanism we proposed in [13].

2) Collaboration of MEC servers (Second part): This part is related to resource allocation and load distribution. Each task L_n will first request computation resources Δ_n on its Local MEC Server, where Δ_n is the minimum required resources for L_n to be completed within its delay tolerance.

$$\Delta_n = d_n/(T_n^{max} - T_n^{tx}) \qquad (6)$$

We assign computation resources to the tasks in ascending order of their delay tolerance. However, there may be some unserved task due to the lack of resources on each server. To this end, we will distribute unserved tasks to the Nearby MEC Servers. We will first consider full task distribution since partial task distribution will generate the extra instruction which requires additional resource and time to be executed. For each instruction, it also has its minimum required resources $\Delta_{n,w}$

$$\Delta_{n,w} = d_{n,w}/(T_n^{max} - T_n^{tx} - T_{n,o}) \qquad (7)$$

The complete process of our heuristic algorithm is shown as Algorithm 1. Also, we estimate the computational complexity in the worst cost of our algorithm. In first part, our algorithm costs $O(N_s)$ for classification and $O(N_s logN_s + N_s \cdot HlogH) = O(HN_s logH)$ for channel allocating in each server s. In second part, the complexity largely depends on the sort operation of load distribution, which is $O(NlogN), N = \sum_s N_s$. Hence, the total complexity is $O(N + NHlogH + NlogN) = O(NHlogH)$.

IV. SIMULATION

In simulation, we evaluate our proposed algorithm in a multi-server MEC system consists of 5 MEC servers. The related parameters are listed in Table I at great length. Besides, we also evaluate three other schemes, Local Execution (LE), Remote Execution (RE), and No Load Distribution (NLD) in this section. LE and RE are the schemes that do not consider offloading decision. One always executes tasks on local devices (LE), and the other always offloads tasks to the MEC servers. In addition, NLD is the algorithm we developed in [13] which is designed for single-server scenario and hence does not take load distribution into the discussion.

TABLE I. RELATED PARAMETERS

Parameters	Values	Parameters	Values
B	1×10^6 Hz	σ	-100 dB
F_s	$[10,50] \times 10^9$	W_n	[2,4]
$g_n^{h_n}$	[4,10] dB	a_n	$[100,1000] \times 10^3$
f_n	$[1.5,2.5] \times 10^9$	d_n	$[100,2000] \times 10^6$
P_n^l	[5,10] W	T_n^{max}	[0.1, 1] seconds

In Fig. 2., we evaluate all the algorithms under the different maximal number of mobile devices per MEC server. The number of mobile devices assigned to each MEC server follows the uniform distribution $unif(x-10, x)$, where x is the maximal number. In Fig. 2., we can observe that the resources are limited, so execution cost will increase with the maximal number of mobile devices. Furthermore, this result can be analyzed in two phases. In the first phase, it can be figured out that RE has the worst performance since both radio resources and computation resources are insufficient to run all the tasks on MEC servers. In addition, the curve of LE is nearly a horizontal straight line because all tasks are executed on local devices and hence there is no resources competition problem. Without considering offloading decision, LE and RE lead to poor QoS for end users. In the second phase, we compare our algorithm with the algorithm without taking load distribution into account (NLD). We can observe that our algorithm reduces execution cost with the collaboration of MEC servers and load distribution. Moreover, the result shows that the performance gap between our algorithm and NLD is much larger when the maximal number is around 20 to 70. The reason is that every server can serve its own tasks when the maximal number is small, and otherwise every server has no more resource to serve the tasks from other servers. Therefore, the performance of our algorithm will get close to NLD. On the other hand, load distribution will be performed and thus enhances QoS as the maximal number is in the intermediate number.

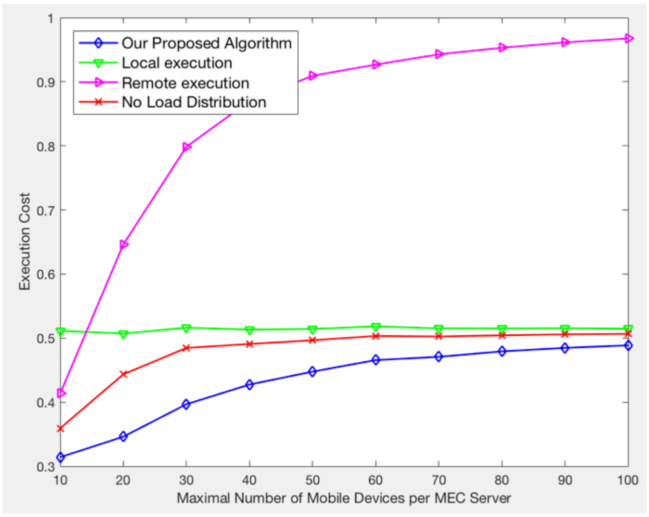

Fig. 2. Execution cost under different schemes versus maximal number of mobile devices per MEC server.

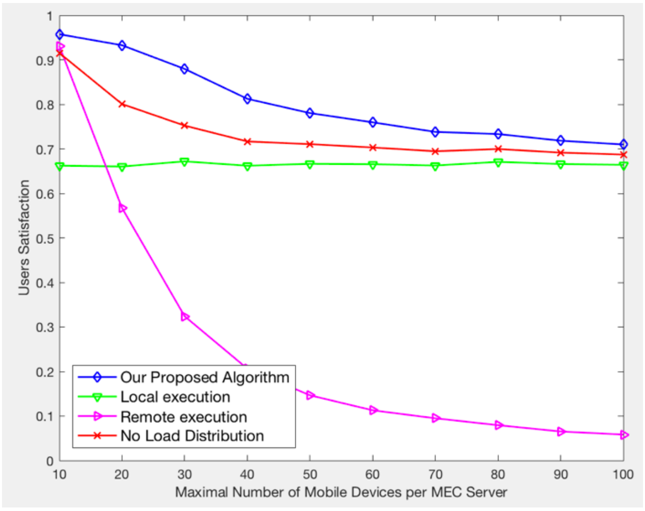

Fig. 3. Users satisfaction under different schemes versus maximal number of mobile devices per MEC server.

We discuss users satisfaction, which means the percentage of tasks completed within the delay tolerance, in Fig. 3. Also, the users satisfaction of LE and RE is much lower than the other two algorithms since they both do not leverage the computation resources on the other side. Besides, the users satisfaction of NLD sharply declines as the maximal number of mobile devices increases. On the other hand, our algorithm offloads some tasks from the server with inadequate resources to the other servers with sufficient resources, and hence the users satisfaction decrease slowly than the other algorithms.

In Fig. 4, the energy consumption of Nearby MEC Servers from viewpoint of MEC Server 1 is discussed. That is, we analyze the energy consumed by the tasks which are distributed from MEC Server 1 to other servers. We assume that it takes 1J for MEC Server to execute 1×10^9 CPU cycles, and hence the energy consumption can be obtained according to the required CPU cycles of tasks. Our algorithm is compared with another algorithm called Aggressive Load Distribution (ALD). In ALD, Local MEC Server does not have higher priority to execute the tasks and, otherwise, the tasks will be frequently distributed to other MEC servers. The result shows that our algorithm consumes less energy. It is because tasks will tend to execute on Local MEC Server rather than on Nearby MEC Servers in our algorithm. With this characteristic, our algorithm can reduce the energy consumption of Nearby MEC Servers.

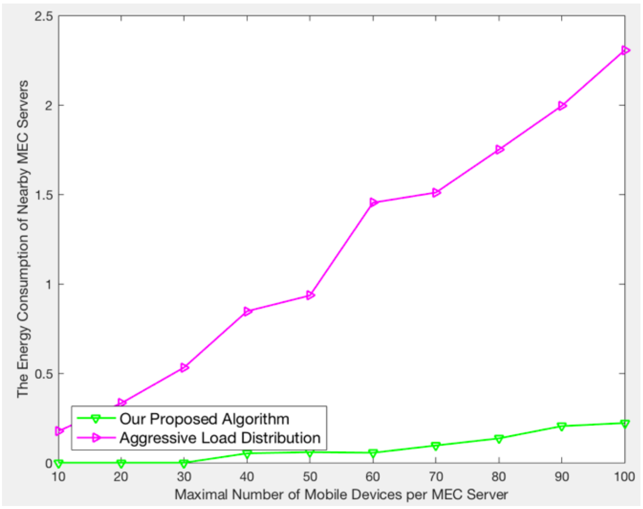

Fig. 4. Energy consumption of Nearby MEC Servers under different schemes versus maximal number of mobile devices per MEC server.

V. CONCLUSION

In this paper, we investigate the optimization of resource allocation and offloading decision. By taking two-dimension resource allocation and offloading decision into consideration, our algorithm will be closer to the optimal usage of the MEC system. Moreover, we improve our previous work from single-server scenario to multi-server scenario and then introduce load distribution into our MEC system. With load distribution, our new method significantly enhances the QoS of end users. As a matter of fact, most of the MEC system will be composed of several servers in the future. Hence, our algorithm can be perfectly performed in the practical system. Furthermore, our load distribution can also save backhaul capacity and energy consumption of MEC servers. The simulation results prove that our algorithm promotes QoS greatly and achieves better performances over other schemes. In addition, in comparison with the scheme without considering load distribution, our algorithm can serve more tasks within the delay tolerance and thus increase users satisfaction.

Acknowledgment

The research is partly funded by Foxconn, MOST of Taiwan under grant 106-2923-E-002-015-MY3, 107-2923-E-002-006-MY3, and 105-2221-E-002-014-MY3, and by the EU Horizon 2020 grant No 761745.

Appendix

A. Local execution

We can obtain the local execution time T_n^l and the local energy consumption E_n^l of the task L_n as

$$T_n^l = d_n / f_n \tag{8}$$

$$E_n^l = P_n^l \cdot T_n^l \tag{9}$$

Where P_n^l is the local computation power of the device n.

B. Wireless Communication

We study a multi-channel system that each BS offers H channels for mobile devices to transmit data. Therefore, we have channels set $\mathcal{H}_s = \{\hbar_s^1, \hbar_s^2, \dots, \hbar_s^H\}$ at each BS s and each mobile device which desires to offload its task has channel selection $h_n \in \mathcal{H}_s$. According to the channel selection, the data rate can be written as

$$r_n^{h_n} = B log_2 \left(1 + \frac{P_n \cdot g_n^{h_n}}{I_{h_n} + \sigma}\right) \tag{10}$$

Where B is the channel bandwidth, P_n is the transmission power of mobile device n, and $g_n^{h_n}$ is the channel gain via the channel h_n. Additionally, I_{h_n} and σ denote the interference and noise respectively. The transmission time T_n^{tx} and transmission energy consumption E_n^{tx} of the task L_n can be obtained as

$$T_n^{tx} = a_n / r_n^{h_n} \tag{11}$$

$$E_n^{tx} = P_n \cdot T_n^{tx} \tag{12}$$

C. Previous method

1) Task Classification and Priority Determination: On each server, we classify $\mathcal{D}_s$ into two sets, $S_{mec,s}$ and $S_{other,s}$ and assign the priority $\emptyset_n$to each device.

$$\emptyset_n = \frac{f_n}{\sum_{i \in D_s} f_i} + \frac{a_n}{\sum_{i \in D_s} a_i} \tag{13}$$

2) Channel Assignment and Resources Allocation for $S_{mec,s}$: We assign the channels to the devices belong to S_{mec} in an ascending order of their $\emptyset_n$ values. Then, we will allocate resources to them by adopting Lagrange Multiplier. The dual function and the result can be written as

$$\min_{y>0} \sum_{n \in \mathcal{D}_s} \frac{\beta_n d_n}{y_n^s} + \lambda \left(\sum_{n \in \mathcal{D}_s} y_n^s - F_s\right) \tag{14}$$

$$y_n^s = \left(\sqrt{\beta_n d_n} / \sum_i \sqrt{\beta_i d_i}\right) \cdot F_s \tag{15}$$

3) Offloading Decision for $S_{other,s}$: For each device belongs to $S_{other,s}$, we also assign the channel and allocate resource to it according to the above mechanism. After that, we will calculate the total execution cost. If the execution cost decreases, this device will select to offload. Otherwise, it will choose local execution.

References

[1] P. Mach and Z. Becvar, "Mobile Edge Computing: A Survey on Architecture and Computation Offloading", *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628-1656, 2017.

[2] Y. Wang, M. Sheng, X. Wang, L. Wang and J. Li, "Mobile-Edge Computing: Partial Computation Offloading Using Dynamic Voltage Scaling", *IEEE Transactions on Communications*, pp. 1-1, 2016.

[3] D. Lin, Y. Hsu and H. Wei, "A Novel Forwarding Policy under Cloud Radio Access Network with Mobile Edge Computing Architecture", *2018 IEEE 2nd International Conference on Fog and Edge Computing (ICFEC)*, 2018.

[4] O. Munoz, A. Pascual-Iserte and J. Vidal, "Optimization of Radio and Computational Resources for Energy Efficiency in Latency-Constrained Application Offloading", *IEEE Transactions on Vehicular Technology*, vol. 64, no. 10, pp. 4738-4755, 2015.

[5] Y. Mao, J. Zhang, S. Song and K. Letaief, "Power-Delay Tradeoff in Multi-User Mobile-Edge Computing Systems", *2016 IEEE Global Communications Conference (GLOBECOM)*, 2016.

[6] X. Chen, L. Jiao, W. Li and X. Fu, "Efficient Multi-User Computation Offloading for Mobile-Edge Cloud Computing", *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795-2808, 2016.

[7] X. Lyu, H. Tian, C. Sengul and P. Zhang, "Multiuser Joint Task Offloading and Resource Optimization in Proximate Clouds", *IEEE Transactions on Vehicular Technology*, vol. 66, no. 4, pp. 3435-3447, 2017.

[8] H. Le, H. Al-Shatri and A. Klein, "Efficient resource allocation in mobile-edge computation offloading: Completion time minimization", *2017 IEEE International Symposium on Information Theory (ISIT)*, 2017.

[9] C. You, K. Huang, H. Chae and B. Kim, "Energy-Efficient Resource Allocation for Mobile-Edge Computation Offloading", *IEEE Transactions on Wireless Communications*, 2017.

[10] S. Sardellitti, S. Barbarossa and G. Scutari, "Distributed mobile cloud computing: Joint optimization of radio and computational resources", *2014 IEEE Globecom Workshops (GC Wkshps)*, 2014.

[11] J. Oueis, E. Strinati and S. Barbarossa, "The Fog Balancing: Load Distribution for Small Cell Cloud Computing", *2015 IEEE 81st Vehicular Technology Conference (VTC Spring)*, 2015.

[12] L. Yang, B. Liu, J. Cao, Y. Sahni and Z. Wang, "Joint Computation Partitioning and Resource Allocation for Latency Sensitive Applications in Mobile Edge Clouds", *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*, 2017.

[13] T. Kan, Y. Chiang, and H. Wei, "Task Offloading and Resource Allocation in Mobile-Edge Computing System", *2018 27th Wireless and Optical Communication Conference (WOCC)*, 2018.