Task Offloading and Resource Allocation in Mobile-Edge Computing System

Te-Yi Kan, Yao Chiang, Hung-Yu Wei
Dept. of Electrical Engineering
National Taiwan University
Taipei, Taiwan
b03901083@ntu.edu.tw

***Abstract*—Mobile-edge computing (MEC) system is a new paradigm to provide cloud computing capacities at the edge of radio access network (RAN) which is close to mobile users. In this paper, we aim to promote QoS by offloading the computationally intensive tasks to the MEC server. There are many papers discuss this issue. Nevertheless, most of them just think over one-dimension resource allocation, radio resources or computation resources, and make the MEC system less effective. Hence, we consider the allocation of both radio resources and computation resources of the MEC server to increase system effectiveness. Apart from this, we take the variety of tasks' requirements into account. That is, we assume that different tasks may have different delay requirements. We formulate this problem as a cost minimization problem and design a heuristic algorithm to address it. Numerical results show that our algorithm can greatly promote QoS.**

Keywords—mobile-edge computing, computation offloading

I. INTRODUCTION

As mobile devices become more and more popular, there are lots of mobile applications emerging and attracting great attention. Most of these applications such as face recognition, interactive gaming and augmented reality require intensive computation capacity and high energy consumption. However, mobile devices are typically resources-constrained. Their computation resources and battery life are limited due to their small physical size. Mobile Cloud Computing (MCC) is the traditional architecture to address these challenges. But offloading tasks to the remote cloud would always cause large delay. Consequently, Mobile-Edge Computing (MEC) is a new paradigm proposed to address this challenge as well as meet the delay requirements of applications. For instance, in [2], the authors assumed that a task can be further divided into some small subtasks. Thus, mobile devices can offload a portion of subtasks of a task and leave the others executing locally. They designed a mechanism to minimize the weighted sum of devices' energy consumption as well as keep the execution delay lower than the delay requirement. [3] introduced 5G heterogeneous networks into the MEC system. In this network, there is a Macro Base Station (MBS) equipped with an MEC server and a Small Base Station (SBS). The service areas of these two stations are overlaid. Hence, mobile devices can offload tasks to not only the MBS but also the SBS. However, only the MBS is equipped with the MEC server, so the tasks offloaded to the SBS should be further transmitted to the MBS incurring backhaul transmission delay. The authors proposed an energy-efficiency mechanism with joint optimization of offloading decision and communication resource allocation. Nonetheless, radio resources between devices and the MEC server and computation resources of the MEC server are both limited. Therefore, allocation of these two types of resources must be included in our discussion. Fog-RAN, such as [4] and [5], are widely used to optimize the services. [4] introduced virtualization of the resources on BBU pool. Then, the authors divided the virtualized resources into two sets, baseband processing and Fog computing. Several experiments were performed on a General Purpose Processor (GPP) in [4]. The authors in [5] considered a multiple F-RAN nodes scenario. They distributed some computing-intensive tasks to these nodes and studied the tradeoff among performance, comping cost and communication. Nevertheless, most of the recent papers only consider one type of resource allocation. Some only consider radio resource allocation. For example, [6] assumed that the total uplink and downlink data rate are fixed value. Each user selects to offload task would allocate the fraction value of the data rate. In [7], there are M wireless channels between users and the base station. Each user selects to offload its task should pick one channel to transmit data. The authors formulate this decision problem as a potential game. But both of these two papers assumed that computation capacity allocated to each user is always the same fixed value. Some only consider computation resource allocation. [8] used Lagrangian multiplier to allocate computation resources to each offloaded task. It also modeled the wireless environment as a multichannel system. Nevertheless, each channel is identical and orthogonal, so there is no radio resource allocation problem. Considering only one resource allocation is less effective. Hence, we take both radio resource and computation resource allocation into consideration. Moreover, most of the recent papers that intend to minimize delay do not consider that different tasks may have different delay requirements. For example, in [9], a task scheduling policy for single-user systems to minimize average delay was developed. The authors assumed that there are many tasks at the mobile device and the device has one computation unit (CU) and one transmission unit (TU). The task can be scheduled to either CU or TU when the unit is idle. If both units are busy, the task would be stay in the queue. They proposed a one-dimensional search algorithm to find the optimal task scheduling policy which minimizes the average delay in power-constrained by adopting Markov decision process. To be

978-1-5386-4959-6/18/$31.00 ©2018 IEEE

more consistent with reality, we aim to minimize the task completion time and take different delay requirements of different tasks into the discussion. We propose a heuristic algorithm to address this offloading decision and resource allocation problem. In our algorithm, we will assign priority to each device according to their local computation frequency and input data size, and the device with higher priority will have precedence to occupy channel and offload the task. By this way, we can let those devices which have more performance gain with offloading to offload their tasks in priority.

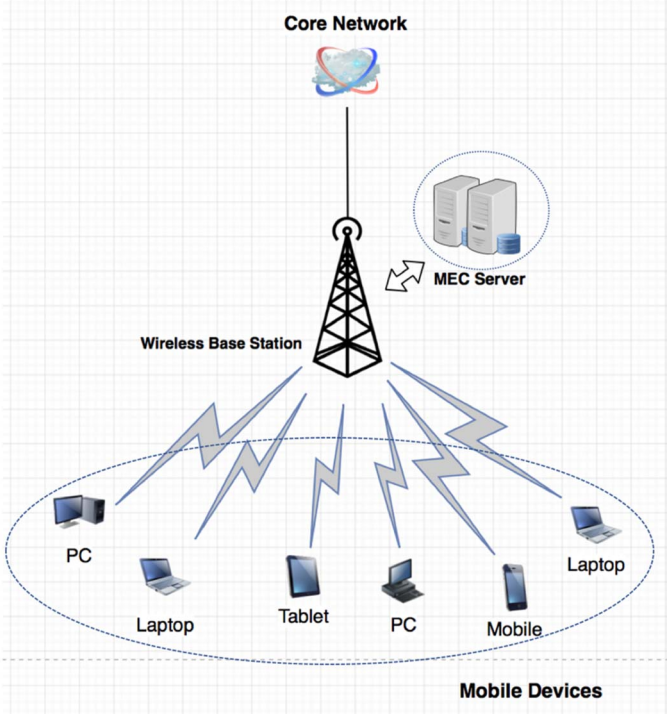

Fig. 1. System architecture

II. PROPOSED METHOD

In this section, we will introduce our multiuse MEC system and formulate the problem. Further, we will propose our heuristic algorithm to address this problem.

A. System Model

We consider a multiuser MEC scenario shown in Fig. 1. In this scenario, there is a single wireless base station (BS) equipped with a MEC server with limited computation capacity F_C and this BS serves N mobile devices. Each device n with remaining energy E_n^R has one task L_n, which can be described in terms of input data size a_n, CPU cycles required d_n and delay tolerance T_n^{max}. Each task can be executed locally or offloaded to the MEC server.

1) Local Execution: We denote computation capacity of device n as f_c (cycles/s) and T_n^l is the local task completion time which can be written as

$$T_n^l = d_n / f_n \tag{1}$$

The energy consumption of device n, E_n^l is

$$E_n^l = P_n^l \cdot T_n^l \tag{2}$$

Where P_n^l denotes the power of device n.

2) Remote Execution: There are M channels between devices and the BS and each of them can be occupied by only one device. The data rate of device n that offloads the task via channel m can be formed as

$$r_n^m = W log_2(1 + SINR_n^m) \tag{3}$$

Where $SINR_n^m = \frac{P_n \cdot g_n^m}{\varpi + I_m}$ and W is the channel bandwidth. Further, P_n is the transmit power of device n and g_n^m is the channel gain between device n and the BS via channel m. In addition, ϖ and I_m denote the noise and the interference of channel m respectively. y_n is denoted as the computation capacity allocated from the MEC server of device n. Hence, the remote task completion time can be written as

$$T_n^r = \frac{a_n}{r_n^m} + \frac{d_n}{y_n} \tag{4}$$

The energy consumption of device n, E_n^r is

$$E_n^r = P_n \cdot \frac{a_n}{r_n^m} \tag{5}$$

B. Problem Formulation

We intend to minimize the completion time of each task as well as considering that different tasks may have different delay requirements. Hence, we define the cost function of device n as

$$C_n = \begin{cases} \frac{T_n}{T_n^{max}} \; if \; T_n < T_n^{max} \; and \; E_n < E_n^R \\ \varphi \end{cases} \tag{6}$$

Where φ is a constant cost for the task whose completion time exceeds the delay tolerance or energy consumption exceeds the remaining energy. In addition, if device n selects to execute locally, $T_n = T_n^l$ and $E_n = E_n^l$. On the other hand, $T_n = T_n^r$ and $E_n = E_n^r$ if device n offloads its task. We formulate a cost minimization problem as follow:

$$\begin{aligned} &\min C = \sum\nolimits_n C_n \\ &s.t. \; C1\text{: } 0 \le y_n \; \forall n \in \mathcal{N} \\ &C2\text{: } \textstyle\sum_n y_n \le F_c \end{aligned} \tag{7}$$

Where constraint $C1$ states that the computation capacity allocated from the MEC server of device n must be a positive value; Constraint $C2$ states that the total capacity allocated from the MEC server must not exceed the computation capacity of the MEC server F_c.

Algorithm 1 The Algorithm for Devices Classification and the Device Priority Determination

Initialization:
  Mobile devices set: N;
  Classified devices sets: $S_{local} = S_{mec} = S_{other} = \emptyset$;
  Priority set: $\Phi = \emptyset$;
for Each device $n \in N$ **do**
  Calculate T_n^l and E_n^l;
  Recieve the information of each channel from the MEC server **and** Calculate r_n^m for each channel;
  if $(T_n^l > T_n^{max})$ **or** $(E_n^l > E_n^R)$ **then**
    Calculate priority ϕ_n according to (8);
    $\phi_n \Rightarrow \Phi$
    $n \Rightarrow S_{mec}$
  else
    if $\max_m\{r_n^m\} \leq \frac{a_n}{\min\{T_n^l, \frac{E_n^R}{P_n}\}}$ **then**
      $n \Rightarrow S_{local}$
    else
      Calculate priority ϕ_n according to (8);
      $\phi_n \Rightarrow \Phi$
      $n \Rightarrow S_{other}$
    end if
  end if
end for
Output: $S_{local}, S_{mec}, S_{other}$ and Φ

C. Algorithm

We design a heuristic algorithm to solve this cost minimization problem in two stages.

1) Device Classification and Priority Determination: In the first stage, we classify devices into three different sets, S_{mec}, S_{local} and S_{other} and assign the priority to each device in S_{mec} and S_{other}. The priority function is shown as follow:

$$\emptyset_n = \frac{f_n}{\Sigma_n f_n} + \frac{a_n}{\Sigma_n a_n} \tag{8}$$

Where the device with the less $\emptyset_n$ value has the higher priority to occupy the channel and offload its task. The complete process is illustrated in Algorithm 1.

2) Channel Assignment and Computation Resource Allocation: In the second stage, we will first check whether the number of devices in S_{mec} is larger than the number of channels. If the number of devices in S_{mec} is larger, we would terminate the tasks which require more computation resources. The terminated task would be regarded as failed and result in the cost value φ. Then, we would assign the channel with highest SINR to each device in S_{mec}. Further, if there are some channels unoccupied, we would assign the channel to the device in S_{other} and estimate the cost C. The devices in S_{other} would select to offload its task only if the cost decreases. Moreover, we introduce Lagrangian multiplier to allocate computation resources to those offloaded tasks as follow:

$$L(y_n, \lambda) = \min_y \Sigma_{n \in S} \frac{T_n^r}{T_n^{max}} + \lambda(\Sigma_{n \in S} y_n - F_c) \tag{9}$$

Where S is the set of offloaded tasks. The complete process is illustrated in Algorithm 2.

Algorithm 2 The Algorithm for Assigning Channels and Allocating Computation Resource

Inputs:
  $S_{local}, S_{mec}, S_{other}$ and the number of channels M;
Initialization:
  Offloading device set $S_{offloading} = \emptyset$;
  total execution cost $C = 0$;
  Remaining number of channels $M^R = M$
if the number of devices in $S_{mec} > M$ **then**
  Remove the top k devices consuming largest computation resources, where $k = M -$ the number of devices in S_{mec};
end if
while $S_{mec} \neq \emptyset$ **do**
  Assign the channel to the device n, where $n = arg\min_n \phi_n$, $n \in S_{mec}$;
  $n \Rightarrow S_{offloading}$
  $S_{mec} \leftarrow S_{mec} \backslash \{n\}$
end while
Allocate computation resources and Calculate the total execution cost C
while there is a device $n \in S_{offloading}$ s.t. $T_n > T_n^{max}$ **do**
  Remove the device i, where $i = arg\max_i y_i$, $i \in S_{offloading}$ and $T_i > T_i^{max}$;
  $S_{offloading} \leftarrow S_{offloading} \backslash \{i\}$
  Allocate computation resources and Calculate the total execution cost C
end while
while $S_{other} \neq \emptyset$ and $M^R > 0$ **do**
  Assign the channel to the device n, where $n = arg\min_n \phi_n$, $n \in S_{other}$;
  Allocate computation resources and Calculate the total execution cost C';
  if $C' > C$ **then**
    $n \Rightarrow S_{local}$
  else
    $n \Rightarrow S_{offloading}$
    $C \leftarrow C'$
  end if
  $S_{other} \leftarrow S_{other} \backslash \{n\}$
end while
Output: S_{local} and $S_{offloading}$

III. Simulation

In our simulations, we set the related parameters as follow. There are 30 devices in our system and 15 wireless channels with bandwidth $W = 1 \times 10^6$ Hz between devices and the MEC server. For each device n, the local computation capacity follows uniform distribution $f_n \in [1.5, 2.5] \times 10^9$ cycles/sec. For each task, the input data size is $a_n \in [100, 1000] \times 10^3$ bits; the number of CPU cycles needed is $d_n \in [100, 1000] \times 10^6$ cycles, and the delay tolerance is $T_n^{max} \in [0.2,\ 1]$ sec. Besides, the total computation capacity of the MEC server $F_c = 100 \times 10^9$ cycles/sec.

We compare the performance of our proposed algorithm with the following two schemes:

- Always Local Execution: tasks would always select local execution.
- Always Remote Execution: tasks would always select to execute on the MEC server.

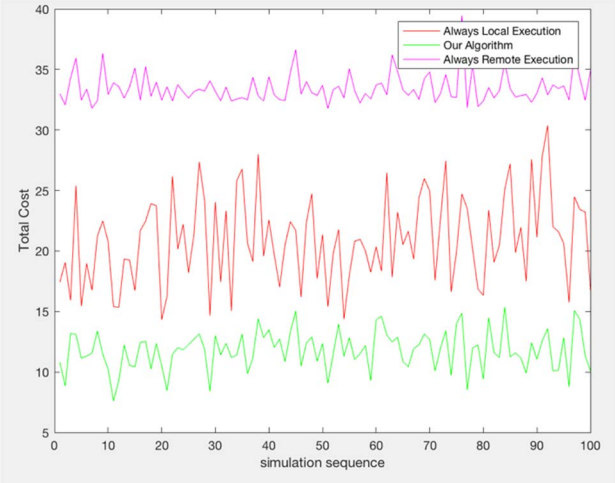

Fig. 2. Total execution cost under different schemes

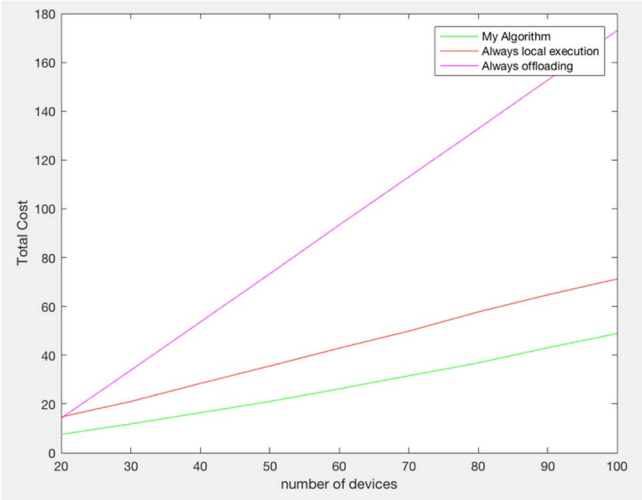

Fig. 3. Total execution cost under different schemes versus the different number of devices

In our first simulation, we randomly generate devices and channels for 100 times and evaluate the performance of each scheme. The results are shown as Fig. 2. Fig. 2 shows that our algorithm outperforms the other two schemes in any time of simulations. It represents that our algorithm would promote QoS in any situation of devices and channels. Further, the Always Local scheme cause the larger value of cost. We can conclude that the completion time is greatly reduced with computation offloading. Besides, Always Remote Execution scheme incurs the much larger cost. That is because the radio resources are limited. If all the devices tend to offload their tasks, some tasks would suffer larger waiting time and their completion time would exceed their delay tolerance.

In Fig. 3, we study the total cost of the system versus the number of devices *N*. We observe that our algorithm leads to the lower value of the total cost than the other two schemes in any case. Furthermore, when the number of devices is large, the curves of all the schemes become linear line. The reason is that the number of channels is limited and each channel can only be assigned to one device. Thus, when the number of devices is much larger than the number of channels, most of the devices can only select to locally execute their task.

IV. CONCLUSION

In this paper, we formulate the multiuser offloading decision and two types of resource allocation problem as a cost minimization problem and proposed a heuristic algorithm considering both radio and computation resource allocation. In addition, we also discuss the variety of delay requirements of tasks in spite of only thinking over the average delay. By taking different delay requirements and two types of resource allocation into account, our algorithm can gain more effectiveness and be closer to reality. In the simulation, we evaluated the effectiveness of our proposed algorithm by comparing with the other two schemes, local execution only and remote execution only. Numerical results show that our proposed algorithm outperforms other two schemes and greatly promote QoS by taking two types of resource allocation into account at the same time. Furthermore, we show that our algorithm leads to the lower value of the total cost than the other two schemes under any condition of the number of devices.

REFERENCES

[1] Y. Jararweh, A. Doulat, O. AlQudah, E. Ahmed, M. Al-Ayyoub and E. Benkhelifa, "The future of mobile cloud computing: Integrating cloudlets and Mobile Edge Computing", *2016 23rd International Conference on Telecommunications (ICT)*, 2016.

[2] C. You, K. Huang, H. Chae and B. Kim, "Energy-Efficient Resource Allocation for Mobile-Edge Computation Offloading", *IEEE Transactions on Wireless Communications*, vol. 16, no. 3, pp. 1397-1411, 2017.

[3] 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.

[4] Y. Ku, D. Lin, C. Lee, P. Hsieh, H. Wei, C. Chou and A. Pang, "5G Radio Access Network Design with the Fog Paradigm: Confluence of Communications and Computing", *IEEE Communications Magazine*, vol. 55, no. 4, pp. 46-52, 2017.

[5] Y.-Y. Shih, W.-H. Chung, A.-C. Pang, T.-C. Chiu, and H.-Y. Wei, "Enabling low-latency applications in fog-radio access networks", *IEEE Network*, vol. 31, no. 1, pp. 52-58, 2017.

[6] M. Chen, B. Liang and M. Dong, "Joint offloading decision and resource allocation for multi-user multi-task mobile cloud", *2016 IEEE International Conference on Communications (ICC)*, 2016.

[7] 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.

[8] 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.

[9] J. Liu, Y. Mao, J. Zhang and K. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems", *2016 IEEE International Symposium on Information Theory (ISIT)*, 2016