IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 15, 2017, accepted November 10, 2017, date of publication November 24, 2017,
date of current version February 14, 2018.

Digital Object Identifier 10.1109/ACCESS.2017.2776323

Competitive Partial Computation Offloading for Maximizing Energy Efficiency in Mobile Cloud Computing

SANGHONG AHN[1], (Student Member, IEEE), JOOHYUNG LEE[2], (Member, IEEE), SANGDON PARK[3], (Member, IEEE), S. H. SHAH NEWAZ[4], (Member, IEEE), AND JUN KYUN CHOI[1], (Senior Member, IEEE)

[1]School of Electrical Engineering, Korea Advanced Institute of Science and Technology, Daejeon 34141, South Korea
[2]Department of Software, Gachon University, Seongnam 461-701, South Korea
[3]Information and Electronics Research Institute, Korea Advanced Institute of Science and Technology, Daejeon 34141, South Korea
[4]School of Computing and Informatics, Universiti Teknologi Brunei, Gadong A BE1410, Brunei Darussalam

Corresponding authors: Joohyung Lee (j17.lee@gachon.ac.kr) and Sangdon Park (sangdon.park@kaist.ac.kr)

This work was supported in part by the Institute for Information and Communications Technology Promotion Grant funded by the Korean Government (MSIT), Development of Trusted Information Infrastructure, S/W Framework for Realizing Trustworthy IoT Eco-System, under Grant 2015-0-00533, and in part by the Basic Science Research Program of the National Research Foundation of Korea under Grant NRF-2017R1C1B5017232.

ABSTRACT In this paper, we newly model computation offloading competition when multiple clients compete with each other so as to reduce energy cost and improve computational performance. We consider two types of destination of offloading request, such as a cloudlet and a remote cloud. Here, the cloudlet consists of locally connected mobile terminals with low-latency and high bandwidth but suffering from task overload due to its limited computational capacity. On the other hand, the remote cloud has a high and stable capacity but the high latency. To facilitate the competition model, on the destination sides, we have designed an energy-oriented task scheduling scheme, which aims to maximize the welfare of clients in terms of energy efficiency. Under this proposed job scheduling, as a joint consideration of the destination and client sides, competition behavior among multiple clients for optimal computation offloading is modeled and analyzed as a non-cooperative game by considering a trade-off between different types of destinations. Based on this game-theoretical analysis, we propose a novel energy-oriented weight assignment scheme in the mobile terminal side to maximize mobile terminal energy efficiency. Finally, we show that the proposed scheme converges well to a unique equilibrium and it maximizes the payoff of all participating clients.

INDEX TERMS Mobile cloud computing, cloudlet, job scheduling, noncooperative game, computation offloading.

I. INTRODUCTION

The demand for multimedia applications, such as video streaming has been growing stupendously over the last several years with innovations in mobile terminals and wireless communication technologies [1]. Recently, those multimedia applications have been evolving to virtual reality (VR) or augmented reality (AR) services, for rich and realistic experience of multimedia contents [2], [3]. Nevertheless, two major limitations of mobile terminals, namely, electric power supply and processing capacity, are the main impediments to the wide-spread use of those advanced multimedia applications. To overcome these two limitations, mobile cloud computing (MCC) [4], [5], which provides resources from the cloud to the mobile terminals, has been steadily gaining importance in the market. It also has various benefits such as reduced cost, easier maintenance, and automatic scaling. Among the MCC architectures, a cloudlet model or multi-tier cloud architecture is an attractive solution. The cloudlet model is a local cloud composed of mobile terminals as a local server. It can reduce the traffic delay and cost of cloud communication because its clients are connected in a local network.

For MCC, to date, various server selection problems at the client and job scheduling problems at the cloud server have been studied in the literature [6]–[9]. In particular, the existing research efforts mostly are aiming at devising

an optimal operation of MCC, taking into account latency, energy efficiency, and task overload at the server. Recently, by adopting computation offloading in MCC, a client can partially offload the computation of applications to the cloud server, and the cloud server improves the overall performance of mobile terminals by allocating its resources as high level granularity of parallelism [10], [11]. Here, how much of the application codes needs to be partially offloaded to the cloud and how to schedule offloaded computations of multiple mobile terminals are challenging. These concerns have recently motivated studies of the computation offloading problem in MCC to enable the more effective utilization of resources at clouds to improve the overall performance of clients [12], [13]. By considering Wi-Fi channel competition from mobile terminals, an opportunistic offloading scheme in MCC was proposed by Chen *et al.* [12]. Specifically, they considered decision making on the computation offloading decision based on the given size of the offloading code. The work of Chen [13] is the closest to our research, where the focus was on studying a competition game by joint consideration between the network status and mobile terminal sides. Here, a mobile terminal finds the optimal size of offloading code to maximize energy efficiency and temporal efficiency. Since the above two works only focused on wireless channel bottlenecks for a remote cloud computing scenario, there was no consideration of task overload problem that can be occurred in a cloudlet scenario.

A task overload of a cloudlet - which means that the computational load is beyond it's capability - is a critical factor of performance. For instance, if many clients determine to offload their computation toward a certain cloudlet simultaneously, the cloudlet would become overloaded. This could result in delaying the task completion time of the previously assigned task, and in turn, this would contribute to worsening the performance of computation offloading. Therefore, it is increasingly important to design a desirable task scheduling scheme for computation offloading in MCC by considering interaction between multiple clients and the cloudlet with consideration of task overload.

In this paper, we model a computation offloading competition when multiple mobile terminals compete with each other to reduce their energy cost and increase the performance. We consider two types of destinations for offloading requests : a cloudlet and a remote cloud. Here, the cloudlet consists of locally connected mobile terminals with low-latency and high bandwidth but suffering from task overload due to its limited computational capacity. On the other hand, the remote cloud has a high and stable capacity but high latency. To devise a computation offloading competition mechanism, in this paper, we design an energy-oriented task scheduling scheme which aims to maximize the social welfare in terms of energy efficiency. This novel job scheduling mechanism takes into account a trade-off between temporal cost and energy gain of the mobile terminals in the proposed computation offloading model. Under this proposed job scheduling, competition behavior among multiple mobile terminals is modeled and analyzed as a non-cooperative game [14]. Based on this game-theoretical analysis, we propose a novel energy-oriented weight assignment scheme in the mobile terminal side, which aims to maximize energy efficiency of mobile terminals. Finally, we demonstrate that our approach converges well to a unique equilibrium, and we show that it also maximizes the utilities of mobile terminals at the equilibrium of the game.

In brief, we make the following contributions in this paper:

- A modeling of partial computation offloading for a target code partition.
- A formulation of the problem of optimal computational resource allocation and utility modeling under multiple offloading requests with the corresponding performance needs and given energy status of clients.
- A task scheduling scheme of the cloudlet considering clients' energy efficiency.
- An algorithm for clients to find a proper strategy under the proposed job scheduling scheme.
- A game-theoretic analysis of client behavior under the proposed solution, and an evaluation of its performance in terms of social welfare and client's energy.

The rest of the paper is organized as follows. Section II reviews previous works on MCC, the load balancing problem and game theoretic approaches. In Section III, we describe our model of cloudlet computing. We then provide details of the proposed task scheduling scheme and the design of utility for entities in a mobile cloud in Section IV. In Section V, we formulate the problem as a non-cooperative game, and provide an analysis of the existence and uniqueness of the equilibrium solution for the game. Numerical results are described in Section VI. Finally, we conclude the paper and discuss directions for future work in Sections VII and VIII.

II. RELATED WORK

Computation offloading, which is widely used for distributed computing, has been considered as a promising solution in MCC. Using computation offloading, the most computationally expensive operations at the code level of mobile terminals can be offloaded for remote processing in MCC. To date, many studies have been conducted in a distributed computing area aiming at improving utilization of resources in clouds and maximizing energy efficiency or application performance in clients [15]–[20]. The major shortcoming of the existing research efforts is that they consider a single client only in their computation offloading solutions. Even these solutions take into account cost models for energy consumption and other temporal factors (e.g. communication medium, network bandwidth and programming model) in decision making process for task load allocation, they cannot be adopted as a realistic solution due to considering only single user in their mathematical formulation (in a MCC, there used to be multiple mobile terminals). In a realistic scenario, a cloudlet of limited servers needs to schedule its computational resources among multiple clients efficiently. It is related to a task overload problem or a job scheduling

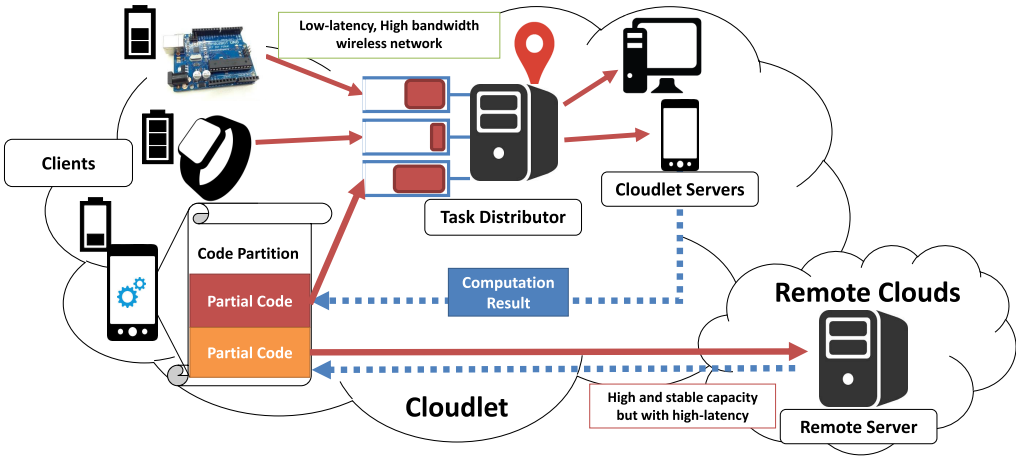

FIGURE 1. Computation offloading model in cloudlet environment.

problem (aka. makespan problem) that is analogous to the optimization of minimizing, which is known as an NP-hard problem. Recently, these issues have been dealt with for a parallel and distributed computing model [21].

The limitation of the research efforts mentioned above has motivated some researchers (e.g. [12], [13], [22]–[26]) to design a job scheduling mechanism for the offloaded code from multiple clients in a MCC. By considering Wi-Fi channel competition from mobile terminals, an opportunistic offloading scheme in MCC is proposed by Chen *et al.* [12]. Specifically, with the given size of offloading code, a decision is made whether to offload or not. Chen [13] proposed a decentralized computation offloading game model where there is a Wi-Fi channel congestion. This work is the closest to our work, in which the focus is to study a competition game based on a joint consideration of the cloud and mobile terminal sides. To the best of our knowledge, this is the only work to date that has suggested joint consideration of the server and client sides. Here, the cloud conducts job scheduling, and a mobile terminal finds the optimal size of offloaded code to maximize energy efficiency. Note that above two works only focused on wireless channel bottlenecks issue when multiple terminals offload code at a remote cloud. In these two works, there was no consideration of task overload problem that can be occurred in a cloudlet scenario. Cardellini *et al.* [27] suggested a game theoretical approach with a queueing network to the task overload problem in a cloudlet environment. However, a task overload problem might occur in the peak time at which multiple clients request offloaded computation simultaneously. In this aspect of view, it is needed to deal with a scheduling policy in a cloudlet computing model.

Therefore, in this paper, we adopt a game-theoretic approach and propose a novel computation offloading scheme with a user centric scheduling discipline for a task distributor and mobile terminals. Specifically, compared with previous studies on computation offloading, we provide a rigorous analytical model to consider the impact of task overload toward clouds. Finally, we expect that this work can contribute to more practical computation offloading design in MCC to improve overall performance of mobile terminals and reduce the task overload problem in the cloudlet systems.

III. SYSTEM MODEL

In this section, we describe the network model and computation offloading process related assumptions for our contribution in this paper. In this study, 'cloudlet' means a local network that consists of multiple mobile terminals and PCs. This network model presented in Fig.1 follows a cloudlet model introduced in [28].

A. CLOUDLET ENVIRONMENT AND TASK OVERLOAD PROBLEM

In our computation offloading model, we assume that there are multiple mobile terminals in a cloudlet; we will call them clients, which are denoted as $C_1, C_2, \ldots, C_N$, where N is the total number of clients. The clients are willing to offload their computation to save their own energy and boost performance. There are two destinations of offloading request: a cloudlet and a remote cloud. We suppose that the cloudlet, which is locally connected mobile terminals or PCs over local wireless network, has low-latency and high bandwidth but suffers from task overload from limited computational capacity. On the other hand, the remote cloud has a high and stable capacity but the high communication latency, which is not negligible.

In the case of offloading to the cloudlet, the clients would not want to suffer delays due to the cloudlet's task overload. Suppose that all concurrent tasks have the same priority. As the number of concurrent tasks increases, the expected completion time of each task also increases. This causes several issues related to task overload; if a task requires a tight deadline, the offloading process cannot satisfy this deadline requirement. Moreover, the energy efficiency of computation offloading might decrease, since the client needs to wait longer time. Accordingly, in clients' side, they can check

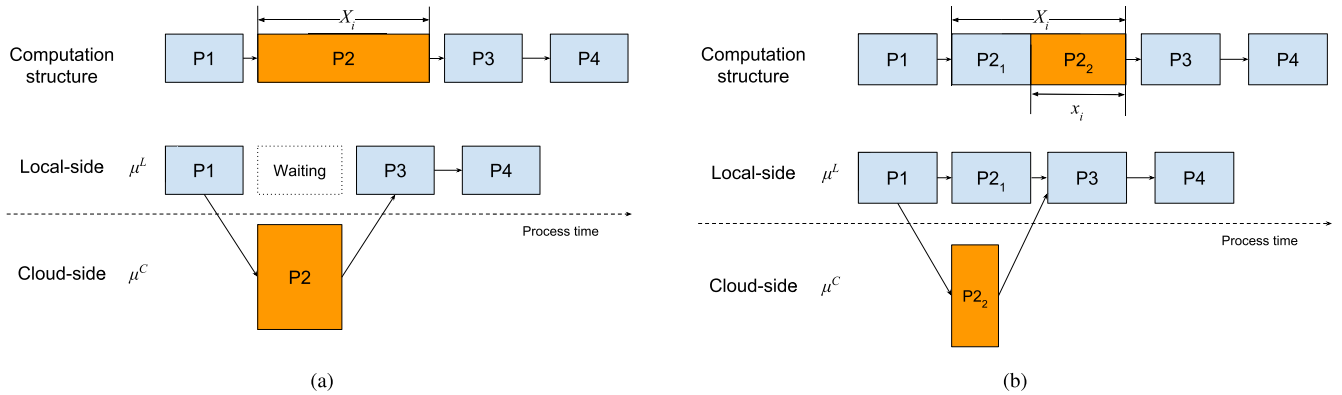

FIGURE 2. Illustration of computation offloading model: (a) opportunistic offloading and (b) parallel partial offloading.

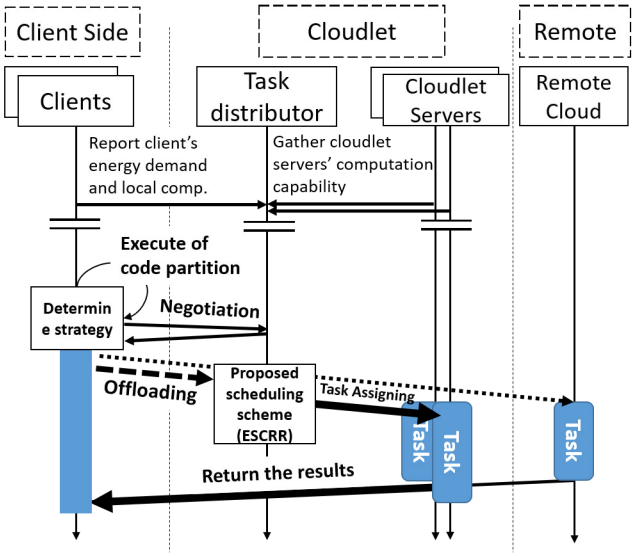

FIGURE 3. A sequential diagram of computation offloading scenario toward cloudlet and remote cloud.

the cloudlet status prior to requesting computation offloading to determine the offloading procedure, according to below offloading modes:

- Opportunistic offloading: as described in Fig.2a, clients decide to offload all of their code partition or not.
- Partial offloading: clients decide to offload a part of their code partition and compute the remainder, as described in Fig.2b

If a code partition can be divided into smaller chunks as in a loop procedure, partial offloading is an attractive approach for clients to improve their performance. Fig.3 describes a sequential diagram of a partial offloading scenario. The procedures of partial computation offloading are described as follows:

1) (Clients & Servers) Registration & Update: First, a task distributor gathers information of clients and servers such as computational capability and energy demands periodically.
2) (Clients) Profiling & code partitioning: Clients divide their own applications' executable code into several partitions and analyze dependencies among the partitions, a critical path, code running time, and their corresponding deadlines. In this step, clients can figure out which partitions can be offloaded. The size of each code partition for the client C_i is determined in this step.
3) (Clients) Run-time offloading negotiation: Clients run their own application. When they identify a remotely executable code, they check the cloudlet status to make an offloading decision. If possible, they determine the size of the offload task. Let the full size of a target code partition for a client C_i be denoted as X_i, while the size of the partial task is denoted as x_i.
4) (Clients) Partial offloading and local computation: Clients send a chunk of executable code and related input data to the task distributor, while they remain busy computing the remains of the code partition. They receive the result from the servers and proceed to the next computation.
5) (Task Distributor) Task scheduling and distribution: Task distributor assigns priorities for requested tasks with its own scheduling scheme. Then, the task distributor allocates tasks to the servers for offloaded computation.
6) Offloaded computation and returning results: Clients gather the results from the cloudlet and remote cloud. Then they return to Step 3 until their application terminates.

In the cloudlet environment, there are multiple mobile terminals which act as cloudlet servers and compute offloaded tasks. A task distributor receives offloading requests from the clients and distributes tasks to the cloudlet servers. The task distributor deals with multiple offloading task by its own scheduling mechanism. There are various scheduling policies to serve multiple jobs. For example, a multi-priority queue and round-robin policy are used for current operating systems. Here, we adopt a multiple priority policy so that the server can adapt a multiple priority policy by assigning a

different computation speed μ_i for each C_i. Weighted round-robin (WRR) scheduling is a policy for serving multiple jobs, such as CPU process handling and packet switching systems [29]. With WRR, multiple jobs are served in round-robin fashion. A WRR scheduler is a preemptive scheduler; in a service cycle, the server computes a small amount of each task. The amount of computation for a task in a single cycle is determined by its priority value such as a weight or a nice value in Linux. Assigning the priority value of a task is also up to the service's policy. For instance, clients might set the priority value for the quality of service, or the distributor could give a higher priority for the shortest task. To guarantee the assigned priorities of offloaded tasks, we assume that tasks can be divided and allocated properly by the task distributor.

We will introduce the design of such a priority-assigning scheme for a cloudlet in Section IV. The algorithmic scheme devised in [30] provides a theoretical framework for the priority-assigning scheme.

In this paper, we discuss a fair job scheduling scheme for whole requests. We do not deal with the details of the code partitioning process; we assume that the components of code partitions such as the size of code blocks and the bandwidth of clients are given.

B. COST AND UTILITY MODELS

Clients expect to save their own energy and improve the performance of applications by computation offloading. This means that there are two kinds of client utility: energy utility and temporal utility.

Many of previous studies have designed a cost model of computation offloading to check the proper offloading time [31]. In this paper, we adopt some cost models from previous studies, and derive a utility model for partial computation offloading.

Before formulating a cost and utility model, we introduce some parameters of clients and cloudlet capacity. Here, μ_i^{local} is the local computation speed of C_i, for which the unit is operations per time. Also, ϵ_i is the energy demand value of C_i. It gets higher when C_i suffers from low energy. We assume that the input data size of computation is proportional to the executable code size. b_i is a transmission time coefficient for C_i, which is regarding the network bandwidth and input data. The total computation capacity of the cloudlet and the remote cloud is denoted as μ_{clet} and μ_{rc}, respectively. Table 1 summarizes a list of symbols used in the paper.

When a client requires offloading, it needs to send executable code and related input data. The data transfer process consumes some energy, and it is proportional to size of the data sent. We formulate the energy cost of computation offloading for C_i, which is denoted as $E_i^{\text{offloading}}$ for communication to cloudlet or remote cloud, as $E_i^{\text{offloading}}(x) = e_i^{\text{tr}} b_i x$, where e_i^{tr} is the energy consumption coefficient during transferring and x is the task size. However, when C_i tries to compute the code partition locally, the energy cost of local computation can be formulated as $E_i^{\text{local}}(x) = e_i^{\text{local}} x$,

TABLE 1. Table of symbols.

Symbol	Meaning
C_i	The i_{th} client
E_i	Energy cost function of C_i
E_i^{gain}	Energy gain function of C_i
T_i^{clet}	Temporal cost of partial offloading to cloudlet for C_i
T_i^{rc}	Temporal cost of partial offloading to remote cloud for C_i
T_i^{local}	Temporal cost of local computation for C_i
X_i	The total size of code partition for C_i
x_i^{clet}	The size of partial task toward cloudlet for C_i
x_i^{rc}	The size of partial task toward remote cloud for C_i
x_i^{local}	The size of local computation task for C_i
s_i	The strategy of $\mu_i^{\text{local}} x_i^{\text{clet}}/(X_i - x_i^{\text{clet}})$ for C_i
μ_{clet}	Total computational capacity of cloudlet
μ_{rc}	Computational capacity of remote cloud
μ_i	Assigned cloudlet speed for C_i
μ_i^{local}	Local computational capacity of C_i
e_i^{local}	A coefficient of energy cost for local computation for C_i
$e_{\text{tr},i}$	A coefficient of energy cost of C_i for transmission
b_i	a transmission time coefficient for C_i
ε_i	A coefficient of energy demand for C_i
κ_i	A handicap factor for C_i
L	The water level in the water-filling algorithm
ω_i	The width of step for strategy s_i
g	A water scaling factor for the water-filling algorithm

where e_i^{local} is energy consumption coefficient during local computation. Now, we formulate the energy cost of partial computation offloading, which is denoted as E_i, as

$$E_i(x^{\text{clet}}, x^{\text{rc}}) = E^{\text{local}}(X_i - (x^{\text{clet}} + x^{\text{rc}})) - E_{\text{offloading}}(x^{\text{clet}} + x^{\text{rc}}),$$

where x^{clet} and x^{rc} is the size of offloading code to the cloudlet and remote cloud, respectively. X_i is the full size of a target code partition for C_i. The energy gain from partial computation offloading can be calculated as following:

$$E_i^{\text{gain}}(x^{\text{clet}}, x^{\text{rc}}) = E_i^{\text{local}}(X_i) - E_i(x^{\text{clet}}, x^{\text{rc}}). \tag{1}$$

We formulate the temporal cost of computation offloading to the cloudlet for C_i, which is denoted as T_i^{clet}, as $T_i^{\text{clet}}(x) = x/\mu_i + b_i x$, where μ_i is the assigned computation speed from task distributor for C_i. The temporal cost of computation offloading to the remote cloud as $T_i^{\text{rc}}(x) = x/\mu_{\text{rc}} + b_i x + R_{\text{rc}}$, where R_{rc} is the expected latency. Finally, the temporal cost of local computation can be formulated as $T_i^{\text{local}}(x) = x/\mu_i^{\text{local}}$. Now, we can formulate the temporal cost of partial computation offloading. If offloading processes and local computation can be done in parallel, the temporal cost of partial computation offloading is determined as the greatest

value among three factors, like

$$T_i(x^{\text{clet}}, x^{\text{rc}}, x^{\text{local}}) = \max\left(T_i^{\text{clet}}(x^{\text{clet}}), T_i^{\text{rc}}(x^{\text{rc}}), T_i^{\text{local}}(x^{\text{local}})\right).$$

By observation, the energy gain increases as the size of the partial task increases no matter how long processes of offloading take. However, we can observe that the temporal cost can be changed with local computation speed and assigned cloudlet computation speed. To evaluate utility of clients in aspects of both energy gain and temporal cost, it is required to see energy efficiency of an offloading process. Now, we define utility of a client with a ratio of the shortest time to the shortest time among non-zero value of T_i^{clet}, T_i^{rc} and T_i^{local}, which is determined as

$$U_i(x_i^{\text{clet}}, x_i^{\text{rc}}) = \epsilon_i \log\left(1 + \frac{T_i^{\text{short}}}{T_i^{\text{long}}}\right), \tag{2}$$

where ϵ_i is a value of energy demand for C_i, $T_i^{\text{short}} = \min(T_i^{\text{clet}}, T_i^{\text{rc}}, T_i^{\text{local}})$ and $T_i^{\text{long}} = \max(T_i^{\text{clet}}, T_i^{\text{rc}}, T_i^{\text{local}})$ for $T_i^{\text{clet}}, T_i^{\text{rc}}, T_i^{\text{local}} > 0$. Since the expected completion time of a code partition is related to T_i^{long}, it is important to make T_i^{long} be smaller for the client's energy efficiency. The greater value of $T_i^{\text{short}}/T_i^{\text{long}}$ is better for the client, because this value indicates a time balance for energy efficiency. Since T_i^{clet} is determined by competition, the time balance is important for clients. In parallel processing, a client needs to wait completing the longest process of partial offloading even if it already has finished other processes. If the time balance gets near to one, it means that all processes of partial offloading would get finished in a similar time. Note that the value of $T_i^{\text{short}}/T_i^{\text{long}}$ cannot be bigger than one. We adopt a logarithm utility for this problem, which is a famous form to describe a characteristic of marginal utility.

IV. PROPOSED SOLUTION

In this section, we present our job scheduling scheme for multiple tasks in a cloudlet. For a multi-priority scheduling, the rule and method of assigning priority is an important issue for fair and efficient processing and to avoid task overload situation. Since many clients in the cloudlet model are mobile terminals, the task distributor can consider the local computation speed, energy status, and energy efficiency of clients for scheduling. It is related to utility of clients. If clients have knowledge of the cloudlet scheduling policy and mechanism, they can determine proper action to maximize their utility. In this situation, max-min fairness is an important property of scheduling for cloudlet's computational resource which is shared by competing demands.

For max-min fair assignment, the shortest remaining job first (SJRF) policy can be considered in the task distributor due to its advantage in partial offloading. Specifically, the task distributor can lead clients to conduct partial offloading with reduced size of requested task in order to avoid the starvation issue. Then, more clients can be accommodated in partial offloading from cloudlets. Nevertheless, this SJRF policy still has some drawbacks such as the starvation issue for a long task. Thus, to compensate the drawbacks of SJRF, we can consider following:

- Considering local computation speed, remained code size and energy demand of client: If the task size is large, it could receive a large amount of resource allocation with client's status. Assume that there are two clients competing for computation resources, and energy status of each client is different; one is abundant and the other one lacks energy. In this case, by allocating more resource to low-energy clients, all users in the system are able to keep using their applications for a longer time. This can be realized by allocating more weight to the request of a client that has smaller residual energy.
- Handicap for high offloading frequency: This policy can prevent clients from dividing their code partitions into smaller chunks and requesting frequently to receive higher computation speed.

The task distributor assigns computation speed μ_i for the requested task size $x_i^{\text{clet}} = x_i$ of client C_i. In the previous section, we discussed the cost and utility of clients by partial computation offloading. The purpose of the proposed scheduling scheme is to maximize the utility of clients. To avoid the problem from getting too complex, we assume that the task distributor of cloudlet have no information of the remote cloud.

To make this argument clearer, we introduce a utility function of partial computation offloading from (2), taking into account the energy status of clients, and applying it to priority assignment. By observation, the utility of a client is related to the expected completion time of a requested task. The factor in the logarithm in (2) can be rewritten as

$$\frac{T_i^{\text{local}}}{T_i^{\text{clet}}} = \frac{\mu_i}{\mu_i^{\text{local}} \frac{x_i}{X_i - x_i}}.$$

We convert the problem into a maximization problem. We can derive a new form of utility as

$$\begin{aligned} U_i'(x_i) &= \omega_i \log\left(\frac{\mu_i}{\mu_i^{\text{local}} \frac{x_i}{X_i - x_i}} + 1\right) \\ &\text{s.t. } 0 < \mu_i^{\text{local}} \frac{x_i}{X_i - x_i} \le \mu_{\text{clet}} \quad \forall i, \\ &\sum_i^N \mu_i \le \mu_{\text{clet}}, \end{aligned} \tag{3}$$

where $\omega_i = \epsilon_i \kappa_i$. To prevent users from using exploits and abusing, we introduce κ_i as a handicap factor for a high frequency and repeated request of high energy demand, which is determined by the system. Note that $0 < \kappa_i \le 1$.

These kinds of policies can be achieved by a water filling algorithm. In the proposed water-filling algorithm, we consider the following:

- The local computing power and the ratio of requested task $\mu_i^{\text{local}} x_i/(X_i - x_i)$ is the step size s_i of the requested task for client C_i. If the step size gets bigger than the

total computational capacity of cloudlet μ_{clet}, then the algorithm will regard it as $s_i = \mu_{\text{clet}}$.

- ω_i is regarded as the width of a step. Note that the value of ω_i needs to be lower-bounded to 1 by proper scaling, to prevent the step height from being too high.
- The height of a step h_i can be formulated as $h_i = \mu_i^{\text{local}} x_i/(\omega_i(X_i - x_i))$.
- The amount of water - in this case, the total computing power of cloudlet - is $g\mu_{\text{clet}}$, where g is a scaling factor.

Fig.4 illustrates the example of the water-filling problem and solution. The pseudo-codes of energy-oriented weight allocation are presented in Algorithm 1. This is called the energy status and local computing power oriented priority allocation scheme (ESCRR).

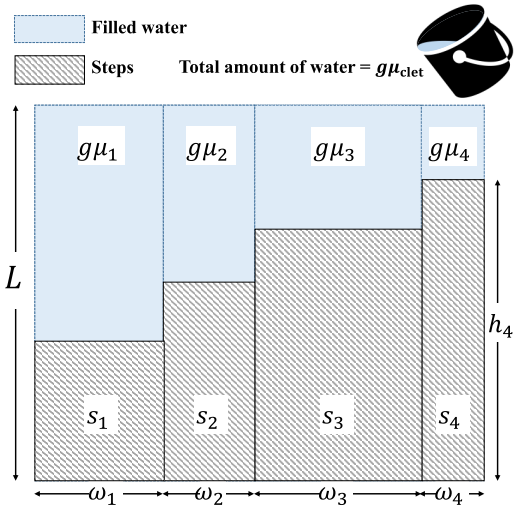

FIGURE 4. Illustration of proposed water-filling scheme.

Based on the proposed Algorithm 1, the task distributor tries to allocate its computation resource by performing a water-filling algorithm.

Theorem 1 (Priority Assignment Problem): There is an optimal allocation of priorities $\mathbf{M} = \{\mu_1, \ldots, \mu_N\}$ *of following optimization problem*

$$\begin{aligned} \max_{\mathbf{M}} \ & \sum_{i=1}^{N} \omega_i \log\left(\frac{\mu_i}{\mu_i^{local}\frac{x_i}{X_i - x_i}} + 1\right) \\ s.t. \ & \sum_{i=1}^{N} \mu_i \le \mu_{clet}, \\ & 0 < \mu_i^{local}\frac{x_i}{X_i - x_i} \le \mu_{clet} \quad \forall i, \\ & 0 \le \mu_i \le \mu_{clet} \quad \forall i. \end{aligned} \tag{4}$$

and the explicit solution of (4) is

$$\mu_i = \frac{L\omega_i - \mu_i^{local}\frac{x_i}{X_i - x_i}}{g}. \tag{5}$$

Algorithm 1 Energy Status and Local Computing Power Oriented Priority Allocation Scheme (ESCRR)

Function ESCRR(P)
  Known: μ_i^{local} as the local computation speed of $C_i, \forall i$,
  ϵ_i as energy status value of $C_i, \forall i$,
  κ_i as a handicap factor of $C_i, \forall i$,
  total computational capacity of cloudlet μ_{clet}
  Input: A sorted list of requests $P = (\mathbf{x}, \mathbf{X})$ where $\mathbf{x} = (x_1, \ldots, x_N)$ and $\mathbf{X} = (X_1, \ldots, X_N)$
  Output: A set of computation speed allocation $\mathbf{M} = < \mu_1, \ldots, \mu_N >$
  /* Determine scale factor g, Step sizes $\boldsymbol{S}$ and width of steps $\boldsymbol{W}$ */
  $g \leftarrow 2N$;
  $\boldsymbol{S} \leftarrow \{s_1, \ldots, s_N\}$
  foreach i **do**
    if $(x_i \le X_i \frac{\mu_{\text{clet}}/\mu_i^{\text{local}}}{1+\mu_{\text{clet}}/\mu_i^{\text{local}}})$ **then**
      $s_i \leftarrow \mu_i^{\text{local}} \frac{x_i}{X_i - x_i}$;
    else
      $s_i \leftarrow \mu_{\text{clet}}$;
    end
  end
  $\boldsymbol{W} \leftarrow \{\omega_1, \ldots, \omega_N\}$ where $\omega_i \leftarrow \epsilon_i \kappa_i$;
  $L \leftarrow \frac{g\,\mu_{\text{clet}} + \sum_{i=1}^{N} s_i}{\sum_{i=1}^{N} \omega_i}$;
  foreach i **do**
    $\mu_i \leftarrow \frac{(L\,\omega_i - s_i)}{g}$;
  end
end

Proof: For convenience, let $s_i = \mu_i^{\text{local}} x_i/(X_i - x_i)$. Then the equation (4) becomes

$$\max \sum_{i=1}^{N} \omega_i \log\left(\frac{\mu_i}{s_i} + 1\right).$$

The function of *ESCRR* gives (5) with a water level of

$$L = \frac{g\mu_{i^*} + s_{i^*}}{\omega_{i^*}}, \tag{6}$$

where $i^* = \arg\max_i \frac{s_i}{\omega_i}$.

This implies that

$$L = \frac{g\mu_{i^*} + s_{i^*}}{\omega_{i^*}} = \frac{g\mu_i + s_i}{\omega_i}, \quad \forall i. \tag{7}$$

Let

$$r = \frac{\omega_{i^*}}{g\mu_{i^*} + s_{i^*}} = 1/L \tag{8}$$

and $\Gamma = \{\gamma_i | \gamma_i = \frac{\omega_{i^*}}{g\mu_{i^*} + s_{i^*}} - \frac{\omega_i}{g\mu_i + s_i}, \forall i\}$. Then we have

$$\gamma_i = 0, \quad \forall i \tag{9}$$

We have the Lagrange function of the constrained optimization problem (4) as

$$G(\mathbf{M}, \Gamma, r) = \sum_{i=1}^{N} \omega_i \log(1 + \mu_i/s_i) - r\left(\sum_{i=1}^{N} g(\mu_i - \mu_{\text{clet}})\right) + \sum_{i=1}^{N} \gamma_i \mu_i. \quad (10)$$

By checking the Karush-Kuhn-Tucker(KKT) condition, the following system holds:

$$\begin{cases} \dfrac{\omega_i}{g\,\mu_i + s_i} - r + \gamma_i = 0, & \forall i \\ \mu_i \geq 0, & \forall i \\ \gamma_i \mu_i = 0, & \forall i \\ \gamma_i \geq 0, & \forall i \\ \sum_{i=1}^{N} \mu_i = \mu_{\text{clet}}, & r \in \mathbb{R}. \end{cases} \quad (11)$$

By observation, it is a differentiable convex optimization problem with linear constraints, which satisfies the KKT conditions (8). Hence, we can conclude that (5) is an optimal solution of the problem. ■

Algorithm 1 modifies the scaling factor k to secure allocations of all tasks. This means that the water level of L that is determined by the algorithm is always higher than the height of the highest step, so that Algorithm 1 can allocate a value that is greater than zero.

Proposition 1: Algorithm 1 gives the optimal solution of the optimization problem of (4).

Note that Algorithm 1 requires $3N$ iterations to compute the optimal solution in the worst case. The worst-case computational complexity is $O(N)$.

The following theorem states that the proposed scheme is desirable and practical.

Theorem 2: The result of ESCRR, which is the set of allocation $\boldsymbol{M}$ is Pareto-optimal.

Proof: The scheme terminates with the fully utilized total resource of μ_{clet}. This implies that we need to decrease the weights allocated to some request to increase the utility of another request. Therefore, the result of Algorithm 1 satisfies the definition of Pareto optimality, which implies that the solution could not be improved without reducing the utility of at least one request. ■

V. GAME THEORETIC ANALYSIS

Clients could estimate the expected completion time for their offloading computation with the distributor's priority assignment. However, when task overload takes place (demand exceeds the total available resource), some tasks may not receive adequate resource allocation from the task distributor, and this could drop clients' utility value. To avoid this, the clients seek to determine the appropriate size of partial offloading task. In this regard, in the light of game-theoretic analysis, we studied the behavior of this type of clients.

A. GAME FORMULATION AND NASH EQUILIBRIUM

In our problem formulation, we assume that the total computational speed of the cloudlet μ_{clet}, the energy status of all clients ϵ_i, and handicap factor κ_i are common knowledge. This enables clients to predict other competing clients' action so that they can determine a strategy as a response to their expectation. Since task distributor does not consider the remote cloud, clients would like to determine the task size toward the cloudlet first. The non-cooperative competition game can be defined as follows:

- Players: Clients which try to request offloading at the same time.
- Strategy: An offloading request $s_i = \mu_i^{\text{local}} x_i/(X_i - x_i)$ for client C_i. The request vector $\boldsymbol{S} = s_1, \ldots, s_N$ is a strategy profile.
- Utility: The players try to maximize their own utilities. In this case, the utility for each player is the energy efficiency against the opportunity cost, which is described in (2). In this game, we assume that players do not consider the remote cloud, which means $T_i^{\text{rc}} = 0$.

From the utility of energy efficiency (2), it is observed that the distributor's priority assignment μ_i is an important factor in determining utility. To observe the distributor's resource allocation which is determined by a client's strategy, we introduce the following definition:

Definition 1: The task distributor's allocation of μ_i for client C_i is formulated as

$$\mu_i = M_i(s_i) = \arg\max_{\mu} \sum_i \omega_i \log\left(\frac{\mu}{s_i} + 1\right),$$

where other players' strategy is fixed as $\boldsymbol{S}_{-i}$.

First, it is necessary to examine properties of defined best-response function.

Lemma 1: The task distributor's priority assignment of $M_i(s_i)$ is strictly decreasing.

Proof: Consider Algorithm 1 of ESCRR. When the size of the partial task increases from zero, the allocated resource of μ_i is decreased because it would cause the increase of its step size. From Algorithm 1, a bigger step size of a client would receive a smaller size of allocation. By observation, decreasing allocation continues until the size of the partial task reaches its maximum size of X_i. ■

In the case of parallel partial offloading, the total processing time is determined as the longer value between an offloading process time and a local computation time. If the local computation time is shorter than the offloading process time, then the utility of energy efficiency is formulated as

$$U_i^{\text{clet}}(s_i) = \epsilon_i \log\left(\frac{M_i(s_i)}{s_i} + 1\right).$$

For the opposite case, the utility of energy efficiency gets formulated as:

$$U_i^{\text{local}}(s_i) = \epsilon_i \log\left(\frac{s_i}{M_i(s_i)} + 1\right).$$

Therefore, the utility of efficiency can be formulated as:

$$U_i(s_i) = \min(U_i^{\text{clet}}(s_i), U_i^{\text{local}}(s_i)).$$

Lemma 2: For client C_i, there is utility function of $u(s_i, \boldsymbol{S_{-i}})$ by the strategy of s_i, where other players' strategies are fixed as $\boldsymbol{S_{-i}} = \hat{\boldsymbol{S}}_{-i}$ and μ_i^{local} and X_i are given. Now, consider two strategies for C_i, $\dot{s}_i$ and $\ddot{s}_i$, satisfying the following conditions:

$$\dot{s}_i < \ddot{s}_i;$$
$$0 < u(\dot{s}_i, \dot{\boldsymbol{S}}_i) = U_i^{clet}(\dot{s}_i) < U_i^{local}(\dot{s}_i).$$

Then, $u(\dot{s}_i, \dot{\boldsymbol{S}}_{-i}) \geq u(\ddot{s}_i, \dot{\boldsymbol{S}}_{-i})$.

Proof: Let $\dot{L}$ be the water level for the strategy set of $(\dot{s}_i, \dot{\boldsymbol{S}}_{-i})$ and let $\ddot{L}$ be that for the strategy $(\ddot{s}_i, \dot{\boldsymbol{S}}_{-i})$, which are given by *ESCRR*. Because $\dot{s}_i < \ddot{s}_i$, once can get $\dot{L} < \ddot{L}$.

First, it will be shown that

$$M_i(\ddot{s}_i) \leq M_i(\dot{s}_i). \tag{12}$$

Suppose that $\ddot{L} - \ddot{s}_i/\omega_i > \dot{L} - \dot{s}_i/\omega_i$. Then, consider the total allocated computational speed to all clients for each strategy of $\dot{s}_i$ and $\ddot{s}_i$, we can get

$$\begin{aligned}\mu_{\text{clet}} &= \frac{\omega_i}{g}\left(\dot{L} - \frac{\dot{s}_i}{\omega_i}\right) + \sum_j \frac{\omega_j}{g}\left(\dot{L} - \frac{s_j}{\omega_j}\right) \\ &< \frac{\omega_i}{g}\left(\ddot{L} - \frac{\ddot{s}_i}{\omega_i}\right) + \sum_j \frac{\omega_j}{g}\left(\ddot{L} - \frac{s_j}{\omega_j}\right) = \mu_{\text{clet}}\end{aligned}$$

which gives a contradiction. Therefore, the inequality condition of (12) is shown.

For the next, if $u(\ddot{s}_i, \dot{\boldsymbol{S}}_{-i}) = U_i^{\text{local}}(\ddot{s}_i) < U_i^{\text{clet}}(\ddot{s}_i)$, then $\ddot{L} - \ddot{s}_i/\omega_i > \dot{L} - \dot{s}_i/\omega_i$ since the offloading process time is shorter than the local computation time. However, this equation gives a contradiction to the inequality condition of (12). This means that $u(\ddot{s}_i, \dot{\boldsymbol{S}}_{-i}) = U_i^{\text{clet}}(\dot{s}_i)$.

By definition of utility when the offloading process time is longer than the local computation time, it is given by

$$U_i^{\text{clet}}(\dot{s}_i) = \epsilon_i \log\left(\frac{M_i(\dot{s}_i)}{\dot{s}_i} + 1\right),$$
$$U_i^{\text{clet}}(\ddot{s}_i) = \epsilon_i \log\left(\frac{M_i(\ddot{s}_i)}{\ddot{s}_i} + 1\right).$$

From the inequality condition of (12), we have

$$U_i^{\text{clet}}(\dot{s}_i) \geq U_i^{\text{clet}}(\ddot{s}_i).$$

■

With Lemma 1 and 2, we can derive following lemma:

Lemma 3: The utility function $u(s_i, \boldsymbol{S_{-i}})\forall i$ is continuous and quasi-concave.

Proof: Since the objective function is continuous and the strategic domain is decreasing, it is clear that the utility function $u(s_i, \boldsymbol{S_{-i}})$ is continuous for all i.

Consider the utility function for client C_i and three strategies of $\hat{s}_i$, $\dot{s}_i$ and $\ddot{s}_i$, satisfying the following conditions:

$$\dot{s}_i < \hat{s}_i < \ddot{s}_i,$$

and fixing the strategies of other clients as $\boldsymbol{S_{-i}} = \hat{\boldsymbol{S}}_{-i}$.

To prove that $u(s_i, \boldsymbol{S_{-i}})$ is quasi-concave, it is necessary to show the following:

$$\min\left(u(\ddot{s}_i, \hat{\boldsymbol{S}}_{-i}), u_i(\dot{s}_i, \hat{\boldsymbol{S}}_{-i})\right) \leq u_i(\hat{s}_i, \hat{\boldsymbol{S}}_{-i}).$$

There are two cases to examine:

Case 1: $0 < U_i^{\text{clet}}(\hat{s}_i) = u(\hat{s}_i, \hat{\boldsymbol{S}}_{-i}) < U_i^{\text{local}}(\hat{s}_i)$

In this case, it is clear that $u(\ddot{s}_i, \hat{\boldsymbol{S}}_{-i}) \leq u(\hat{s}_i, \hat{\boldsymbol{S}}_{-i})$ by lemma 1.

Case 2: $0 < U_i^{\text{local}}(\hat{s}_i) = u(\hat{s}_i, \hat{\boldsymbol{S}}_{-i}) < U_i^{\text{clet}}(\hat{s}_i)$

In this case, $u(\dot{s}_i, \hat{\boldsymbol{S}}_{-i}) \leq u(\hat{s}_i, \hat{\boldsymbol{S}}_{-i})$ since $U_i^{\text{local}}(s_i)$ is a strictly-increasing fuction. ■

With Lemma 3, we can derive following theorem:

Theorem 3: There exists at least one Nash equilibrium (NE) in this game.

Proof: We already show that the strategy set is closed, bounded and convex in Lemma 3. The scheme represents a continuous and concave utility function, by Lemma 3. By [32] for a maximization problem, the non-cooperative game has at least one Nash equilibrium. ■

We know that there exists a Nash equilibrium in this game. By observation, the strategy of $s_i^* = \mu_i^{\text{local}} x_i^*/(X_i - x_i^*)$ will make the utility function u_i as $U_i^{\text{local}} = U_i^{\text{clet}}$. For s_i^*, the value of $M_i(s_i^*)$ becomes

$$M_i(s_i^*) = \mu_i^{\text{local}} \frac{x_i^*}{X_i - x_i^*}.$$

Theorem 4: For any player C_i, the strategy $s_i^* = \mu_i^{local} x_i^*/(X_i - x_i^*)$ is a Nash equilibrium of priority competition game, which is given by

$$s_i^* = h^* \omega_i,$$

where h^* is a number satisfying $\sum_{i=1}^N s_i^* = \mu_{clet}$. The strategy set of $\boldsymbol{S^*} = \{s_1, \ldots, s_N\}$ is a Nash equilibrium point.

Proof: The conditions of $\sum_i s_i = \mu_{\text{clet}}$ and $\sum_i M_i(s_i) = \mu_{\text{clet}}$ gives $M_i(s_i^*) = s_i^*$.

To prove that $\boldsymbol{S^*}$ is a Nash equilibrium point, it is necessary to show

$$u(s_i^*, \boldsymbol{S_{-i}^*}) \geq u_i(s_i, \boldsymbol{S_{-i}^*}),$$
$$\text{where} \quad \forall s_i = \mu_i^{\text{local}} x_i/(X_i - x_i) \quad \forall x_i \in [0, X_i].$$

First, if $s_i \leq s_i^*$, then it gives $u(s_i, \boldsymbol{S_{-i}^*}) \leq u(s_i^*, \boldsymbol{S_{-i}^*})$ by Lemma 1. For the case of $s_i > s_i^*$, suppose that $u(s_i, \boldsymbol{S_{-i}^*}) > u(s_i^*, \boldsymbol{S_{-i}^*})$. Let L^* be the water level in *ESCRR* for strategy $\boldsymbol{S^*}$, and $\bar{L}$ be the water level in *ESCRR* for the strategy of $(s_i, \boldsymbol{S_{-i}^*})$. If $M_i(s_i) > s_i$, then $\bar{L} > L^*$ since $s_i > s_i^*$, which gives

$$(\bar{L} - h^*) \sum \omega_j > h^* \sum \omega_j.$$

However,

$$\begin{aligned}\mu_{\text{clet}} &= s_i + (\bar{L} - h^*) \sum \omega_j \\ &> h^* \sum \omega_j + s_i^* \\ &= \mu_{\text{clet}}\end{aligned}$$

gives a contradiction.

If $M_i(s_i) \leq s_i$, then it should be $s_i^* \leq M_i(s_i^*) < M_i(s_i) \leq s_i$. However, it is observed that $M_i(s_i) \leq M_i(s_i^*)$ when $s_i > s_i^*$, from Lemma 1. By conclusion, the strategy of s_i cannot exist in any of case. ■

Note that the height of steps in the NE case is $h^* = \mu_{\text{clet}}/(\sum_i^N \omega_i)$, so that $\sum_{i=1}^N s_i^* = \sum_{i=1}^N \omega_i \mu_{\text{clet}} / (\sum_j^N \omega_j) = \mu_{\text{clet}}$.

B. STRATEGY OF CLIENTS

In the previous section, we discussed the priority allocation game and its Nash equilibrium. Now, we need to provide a proper mechanism to find NE for clients. If clients know the information of other clients, such as energy demand and request rates, then it would be easier to find their strategies of NE. However, in the dynamic environment of a cloudlet, it is hard to notify all of information to clients before they start to request offloading. In this case, clients can find their NE in iterative offloading requests by adjusting the size of a partial task. Based on the characteristics of *ESCRR* and the desires of clients, it is possible to design an algorithm that converges to the optimal strategies. First, a client can estimate the water level of *ESCRR* with the step size of it's request and allocated computational speed. By observation, we can find that the water level of *ESCRR* is twice the heights of steps for partial tasks with the strategy set of NE.

Algorithm 2 is a mechanism to achieve the NE of a priority allocation game. We suppose that at least the total computational capacity of cloudlet μ_{clet} is well-known information for all of clients. Moreover, it is supposed that the energy demand ϵ_i and handicap factor κ_i would not change during iteration. This algorithm is based on an assumption that the water level of the game is twice the height of properly-proposed steps. The water level can be excluded with the step's height and allocated speed from the previous phase. The algorithm calculates the proper size of computation offloading to make the step height be half of the water level.

C. REQUEST TO REMOTE CLOUD

In Section V-A, we analyzed the competitive offloading game without considering remote cloud. As the result of NE strategy s_i^*, a client gets $T_i^{\text{clet}*} = T_i^{\text{local}*}$. Now, the client considers to request offloading toward a remote cloud additionally for increasing its own satisfaction. To increase the client's satisfaction, another offloading procedure needs to increase the energy gain E_i^{gain} of (1). Because E_i^{gain} is proportional to x_i^{rc}, the client is willing to request offloading toward the remote cloud as many as possible. However, the expected response time of remote cloud offloading T_i^{rc} should not exceed $T_i^{\text{clet}*}$.

With this policy, a client can determine the size of partial task toward a remote cloud as

$$x_i^{\text{rc}} = \arg_x \left(T_i^{\text{rc}}(x) = T_i^{\text{clet}*} \right), \tag{13}$$

and the size of local computation as $x_i^{\text{local}} = X_i - (x_i^{\text{clet}*} + x_i^{\text{rc}})$. If $T_i^{\text{local}}(X_i - x_i^{\text{clet}*}) \geq T_i^{\text{local}}(X_i - x_i^{\text{clet}*})$, (13) gets $x_i^{\text{rc}} = X_i - x_i^{\text{clet}*}$ and $x_i^{\text{local}} = 0$ since there is no merit of local computation.

Algorithm 2 Iterative Algorithm to Find Optimal Strategy

Function `IterativeFindNE` (X_i, s_i^{prev}, μ_i^{prev})
  Known values: μ_i^{local} as local computation speed of C_i,
  μ_{clet} as total computational speed of the cloudlet
  Input: The size of C_i's code partition X_i,
  The previous value of request s_i which is initially zero,
  The allocated computational speed by the previous request, μ_i which is initially zero,
  The number of clients N if it is given
  Output: Client C_i's strategy of x_i
  /* Initialize */ **if** $s_i = 0$ **then**
    if *N is unknown* **then**
      $\alpha \leftarrow 1$;
    else
      $\alpha \leftarrow \frac{1}{N}$;
    end
  end
  if $\mu_{i,prev} \neq 0$ & $s_{i,prev} \neq 0$ **then**
    $\alpha \leftarrow \frac{\mu_i^{\text{prev}} + s_i^{\text{prev}}}{2\mu_{\text{clet}}}$;
  end
  $s_i \leftarrow \alpha \mu_{\text{clet}}$;
  $x_i \leftarrow X_i \frac{\mu_{\text{clet}}/\mu_i^{\text{local}}}{1+\mu_{\text{clet}}/\mu_i^{\text{local}}}$;
  if $x_i > X_i$ **then**
    $x_i \leftarrow X_i$;
    $s_i \leftarrow \mu_{\text{clet}}$;
  end
end

VI. NUMERICAL RESULTS

This section presents numerical results to highlight the effectiveness of the proposed theorems and demonstrate how utility changes when a client's strategy changes under the proposed priority assignment scheme. Before presenting the examples, we introduce a range of variables and parameters. We considered a mobile cloudlet with N clients and one server in the cloudlet for this numerical analysis. Each client C_i has a profile of handicap factor κ_i, energy status ϵ_i, and a local computation speed of μ_i^{local}. At the moment of the offloading procedure, C_i gets the total size of a code partition X_i, and it decides the size of a partial task x_i. To meet the algorithm constraints, the values of κ_i and ϵ_i are adjusted from the original values to make the step width in *ESCRR* bigger than one.

In our simulation, the parameters of the size of target code partition X_i and energy status e_i for client C_i are

generated with uniform distribution, by $600 \leq X_i \leq 1200$ and $1 \leq \epsilon_i \leq 2$. Additionally, we consider that in the cloudlet system, the handicap factors of all clients are the same, which is 1 in this performance analysis. Furthermore, we suppose that the network bandwidth and the coefficient of input data are the same. Note that the calculation of utility in this analysis is based on the initial assignment of the task distributor; this means that the re-assignment procedure after the termination of a task is not reflected in the result.

First, we investigate the behavior of clients under the proposed scheme. We examine the change of allocated speed and utility of energy efficiency determined by C_i's strategy. In this example, the number of clients is set to $N = 5$; the total computational capacity of cloudlet is $\mu_{\text{clet}} = 3000$; the sizes of the target code partitions are $\boldsymbol{X} = \{758, 692, 518, 733, 682\}$; the energy status values are $\boldsymbol{E} = \{1.748, 1.297, 1.627, 1.5451.670\}$ and the local computational speeds are $\boldsymbol{M}_{\text{local}} = \{922, 847, 976, 611, 752\}$. To examine the result of the client C_1's behavior, we assume that other players choose fixed strategies of $\boldsymbol{S}_{-1} = \boldsymbol{S}^*_{-1}$ with $\boldsymbol{x}_{-1} = \{x_2, x_3, x_4, x_5\} = \{255, 201, 359, 312\}$.

As seen in Fig.5, the task distributor's assignment decreases by the size of C_i's requesting task, which verifies Lemma 1. Fig.6 shows that the energy efficiency of C_i has a peak value with variation of C_i's strategy, which verifies the quasi-concavity of utility from Lemma 3. This observation actually verifies Theorem 4 because the peak value of energy efficiency given with x_i is around 317, which can be derived by the strategy of Nash equilibrium. The peak value of time balance can be observed from the peak value of energy efficiency by time, $U_i^c = E_i/T_i$, as depicted in Fig 7.

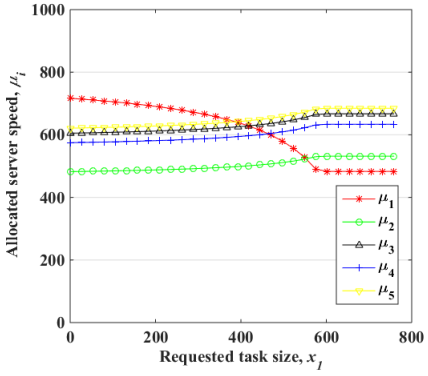

FIGURE 5. Instantaneous computational speed allocation, when $S_{-1} = S^*_{-1}$.

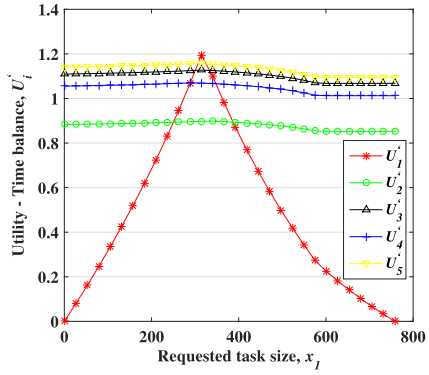

FIGURE 6. Instantaneous utility, when $S_{-1} = S^*_{-1}$.

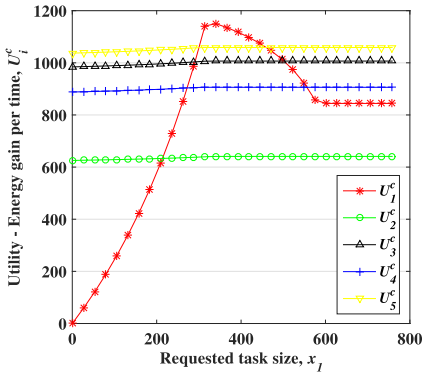

FIGURE 7. Instantaneous utility of energy efficiency, when $S_{-1} = S^*_{-1}$.

Here, we compare the energy efficiency performance of our proposed solution with three scheduling schemes: a WRR scheduler with weight allocation which is proportional to a client's demand (*DWRR*), a non-weighted round robin scheduler (*RR*) and the shortest remaining job first scheduler without regarding other factors (*SJRF*). By comparing social welfare of four scheduling schemes, it is able to figure out the performance of scheduling schemes in aspect of energy efficiency. It is assumed that the allocated weight in *DWRR* and *RR* are $\dot{\mu}_i = \mu_{\text{clet}} s_i / \sum_{j=1}^{N} s_j$ and $\ddot{\mu}_i = \mu_{\text{clet}}/N$, respectively. The social welfare of schedulers is compared, varying the total computation capacity of cloudlet μ_{clet} and N, with two strategy sets, namely the Nash equilibrium and the maximum value set.

In Figs. 8 and 9, we demonstrate that the social welfare can be maximized under our proposed scheme. The simulations have been performed considering $N = 20$. In these simulations, we assume that clients would choose their own NE. Fig.8 shows the comparison of social welfare

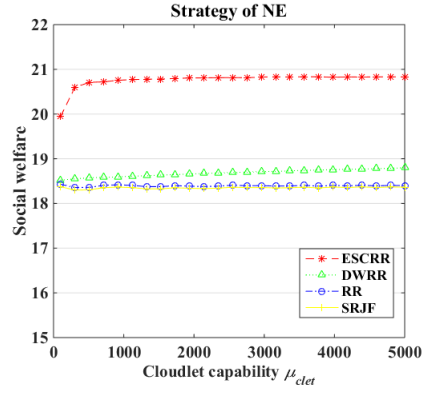

FIGURE 8. Comparison of social welfare, changed by the cloudlet capability.

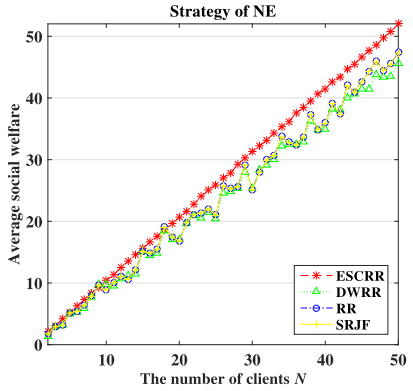

FIGURE 9. Comparison of average social welfare, changed by the number of clients.

that changed by the cloudlet capability μ_{clet}. Fig.9 depicts the average of social welfare for each scheduler under various values of N. In this simulation, the average value for each N is calculated by 100 iterations, where the parameters of clients are generated randomly with uniform distribution. It is important to highlight that, the proposed scheme provides more balanced completion time of cloudlet offloading and local computation than other schedulers, which are related to clients' energy efficiency. Therefore, we can conclude that the proposed scheme generally outperforms the other schedulers in terms of energy efficiency.

Next, we show that Algorithm 2 converges to the Nash equilibrium with acceptable steps of iterations. Fig.10 shows the convergence of utility by the iteration for the case of given N and the case without given N. While iterations, we assume handicap factors of clients do not change for this example. Note that the average number of iterations to converge is gradually increasing with the number of clients, but it is reasonably small by observation. Moreover, it is observed that the average number of iteration to converge gets smaller when the knowledge of N is given to all clients.

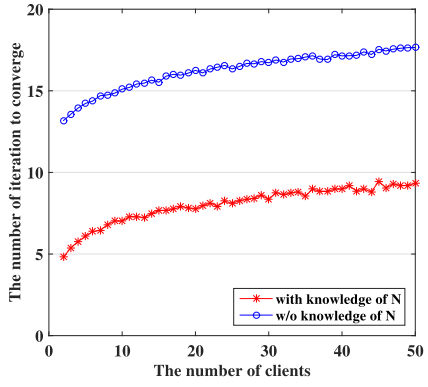

FIGURE 10. Convergence of utility by Algorithm 2.

VII. DISCUSSION ON PRACTICAL ISSUES

In this section, some practical issues of the proposed schemes are discussed. Apart from that, here, we highlight several challenging issues that will be addressed in our future research.

A. TRUSTFUL SCHEME

For a fair resource allocation, a task distributor regards not only the size of requested task but also the subsidiary information of clients such as the energy demand status, client's local computational speed and a total size of original task. This kind of information needs to be generated and sent from client-side. It means that unless all of clients are trustworthy, there could exist a deception of the subsidiary information from clients. Hence, this trustfulness issue of the proposed scheme should be addressed. To solve this issue, in Section III, we already introduced a handicap factor of κ_i in the priority assignment scheme. If client C_i keeps request in improper manner, then the system may assign the value of κ_i smaller for the next request. This kind of policy can prevent clients from abusing or deceiving by introducing penalty. Specifically, the system can give a penalty to the clients who perform as following:

- Local computation speed μ_i^{local} and the total size of original task X_i: These factors affect the total size of step for the proposed water-filling algorithm. If a client deceives μ_i^{local} to be lower or X_i to be higher, then it can choose another strategy to make its utility higher. However, this kind of abusing makes the waiting time of cloudlet offloading be shorter, so that it might cause the client to request in high frequency. To make the value of μ_i^{local} trustworthy, the system may manage this kind of value in the registration stage as a policy. Moreover, it is expected to make clients being honest indirectly by assigning smaller value of κ_i for high-frequency requester.
- Energy demand: the energy demand factor ϵ_i affects the width of step for the proposed water-filling algorithm directly. If a client deceives ϵ_i to be higher, then it can choose another strategy to make its utility higher. To sublate this kind of abusing, the system can consider the progressive handicap which is related to cumulated value of the energy demand. If a client keeps ϵ_i higher than its own, then the progressive handicap gets bigger than normal case and the system may assign the value of κ_i smaller. Therefore, this remedy can reduce the utility of abuser. We remain the issue of periodic initialization of this progressive handicap as the policy of system.

B. IMPLEMENTATION CONCERNS

In Section III, we mentioned a code profiling and partitioning for a process of computation offloading. To implement the proposed system, it is required to argue the level of parallelism and code partitioning methods. For the level of parallelism, data parallelism like MapReduce framework [33] is a good target of the proposed scheme. In data parallelism, a task consists of numerous iterations of small code with data. In this case, the size of offloading task is proportional to the number of iterations. These kinds of offloadable code

partition can be detected in automated profiling system, or be designated by a programmer.

Design of a task distributor is another concern to implement multi-server model of cloudlet offloading system. In the proposed system, a task distributor allocates tasks to the cloudlet servers with proper priorities. It brings a problem about allocating tasks to the servers with ensuring the priority. To solve this problem, it is needed to concern another competition among the cloudlet servers. In this paper, we remain it as an issue of further work and analyze for a single-server scenario. Moreover, evaluation under a dynamic environment with changes of clients' energy demand factor and handicap factor is needed.

VIII. CONCLUSION

In this paper, we proposed an energy efficient job scheduling scheme for offloaded computation in a mobile cloud computing environment. We considered that a task distributor in a cloudlet deals with multiple requests of partial computation offloading with a WRR scheduler. Under this model, we proposed an energy-oriented weight allocation scheme for clients. The proposed scheme assigns a portion of the computational resource for a request, by considering its code size with deadline and energy status. We demonstrated that the proposed scheme can induce the maximal social utility. In our proposed scheme, clients having low residual energy will be guaranteed more utility than those with adequate energy. Here, we proved that a result of weight allocation is Pareto-optimal, and its time complexity is $O(N)$ in a worst case. Moreover, clients try to maximize their utilities by adjusting the size and deadline of offloading code. We also showed that behaviors of clients converge to equilibrium by game-theoretic analysis. It was proven that the equilibrium maximizes utilities of clients. Moreover, we successfully showed that there is a unique Nash equilibrium of this game. In addition to providing a novel solution for mobile cloud computing, this paper presented an insightful discussion as a guide for further research and the deployment of partial computation offloading in a cloudlet environment.

REFERENCES

[1] "Cisco visual networking index: Global mobile data traffic forecast update 2014–2019," Cisco, San Jose, CA, USA, White Paper c11-520S62, 2014.

[2] M. Claypool and K. Claypool, "Latency can kill: Precision and deadline in online games," in *Proc. 1st Annu. ACM SIGMM Conf. Multimedia Syst.*, 2010, pp. 215–222.

[3] W. Zhu, C. Luo, J. Wang, and S. Li, "Multimedia cloud computing," *IEEE Signal Process. Mag.*, vol. 28, no. 3, pp. 59–69, May 2011.

[4] K. Kumar and Y.-H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *Computer*, vol. 43, no. 4, pp. 51–56, Apr. 2010.

[5] S. Kitanov and T. Janevski, "State of the art: Mobile cloud computing," in *Proc. 6th Int. Conf. Comput. Intell. Commun. Syst. Netw.*, 2014, pp. 153–158.

[6] Y. Xu and S. Mao, "A survey of mobile cloud computing for rich media applications," *IEEE Wireless Commun.*, vol. 20, no. 3, pp. 46–53, Jun. 2013.

[7] S. Ray and A. De Sarkar, "Execution analysis of load balancing algorithms in cloud computing environment," *Int. J. Cloud Comput., Services Archit.*, vol. 2, no. 5, pp. 1–13, 2012.

[8] Y. Li and Z. Lan, "A survey of load balancing in grid computing," in *Proc. Int. Conf. Comput. Inf. Sci.*, 2004, pp. 280–285.

[9] A. Carroll and G. Heiser, "An analysis of power consumption in a smartphone," in *Proc. USENIX Annu. Tech. Conf.*, vol. 14. Boston, MA, USA, 2010, pp. 1–14.

[10] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, "A survey of computation offloading for mobile systems," *Mobile Netw. Appl.*, vol. 18, no. 1, pp. 129–140, Apr. 2013.

[11] H. Flores, P. Hui, S. Tarkoma, Y. Li, S. Srirama, and R. Buyya, "Mobile code offloading: From concept to practice and beyond," *IEEE Commun. Mag.*, vol. 53, no. 3, pp. 80–88, Mar. 2015.

[12] M. Chen, Y. Hao, Y. Li, C.-F. Lai, and D. Wu, "On the computation offloading at ad hoc cloudlet: Architecture and service modes," *IEEE Commun. Mag.*, vol. 53, no. 6, pp. 18–24, Jun. 2015.

[13] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 4, pp. 974–983, Apr. 2015.

[14] A. Mas-Colell, M. D. Whinston, and J. R. Green, *Microeconomic Theroy*. London, U.K.: Oxford Univ. Press, 1995.

[15] C. Shi, K. Habak, P. Pandurangan, M. Ammar, M. Naik, and E. Zegura, "COSMOS: Computation offloading as a service for mobile devices," in *Proc. 15th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2014, pp. 287–296.

[16] D. Huang, P. Wang, and D. Niyato, "A dynamic offloading algorithm for mobile computing," *IEEE Trans. Wireless Commun.*, vol. 11, no. 6, pp. 1991–1995, Jun. 2012.

[17] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 945–953.

[18] C. Shi, V. Lakafosis, M. H. Ammar, and E. W. Zegura, "Serendipity: Enabling remote computing among intermittently connected mobile devices," in *Proc. 13th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2012, pp. 145–154.

[19] A. Banerjee, A. Mukherjee, H. S. Paul, and S. Dey, "Offloading work to mobile devices: An availability-aware data partitioning approach," in *Proc. 1st Int. Workshop Middleware Cloud-Enabled Sens.*, 2013, Art. no. 4.

[20] A.-C. Olteanu, N. Tapus, and A. Iosup, "Extending the capabilities of mobile devices for online social applications through cloud offloading," in *Proc. 13th IEEE/ACM Int. Symp. Clust. Cloud, Grid Comput. (CCGrid)*, May 2013, pp. 160–163.

[21] E. Bampis, V. Chau, D. Letsios, G. Lucarelli, I. Milis, and G. Zois, "Energy efficient scheduling of mapreduce jobs," in *Proc. Eur. Conf. Parallel Process.*, 2014, pp. 198–209.

[22] F. Teng and F. Magoulès, "A new game theoretical resource allocation algorithm for cloud computing," in *Advances in Grid and Pervasive Computing* (Lecture Notes in Computer Science), vol. 6104. Berlin, Germany: Springer-Verlag, 2010, pp. 321–330.

[23] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica, "Dominant resource fairness: Fair allocation of multiple resource types," in *Proc. NSDI*, vol. 11. 2011, pp. 1–14.

[24] S. Penmatsa and A. T. Chronopoulos, "Game-theoretic static load balancing for distributed systems," *J. Parallel Distrib. Comput.*, vol. 71, no. 4, pp. 537–555, 2011.

[25] Y. Ge, Y. Zhang, Q. Qiu, and Y.-H. Lu, "A game theoretic resource allocation for overall energy minimization in mobile cloud computing system," in *Proc. ACM/IEEE Int. Symp. Low Power Electron. Design (ISLPED)*, 2012, pp. 279–284.

[26] A. Nahir, A. Orda, and D. Raz, "Workload factoring with the cloud: A game-theoretic perspective," in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 2566–2570.

[27] V. Cardellini *et al.*, "A game-theoretic approach to computation offloading in mobile cloud computing," *Math. Program.*, vol. 157, no. 2, pp. 421–449, 2016.

[28] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *Proc. 1st Ed. MCC Workshop Mobile Cloud Comput.*, 2012, pp. 13–16.

[29] A. Raha, N. Malcolm, and W. Zhao, "Hard real-time communications with weighted round robin service in ATM local area networks," in *Proc. 1st IEEE Int. Conf. Eng. Complex Comput. Syst. (ICECCS)*, Nov. 1995, pp. 96–103.

[30] P. He, L. Zhao, S. Zhou, and Z. Niu, "Water-filling: A geometric approach and its application to solve generalized radio resource allocation problems," *IEEE Trans. Wireless Commun.*, vol. 12, no. 7, pp. 3637–3647, Jul. 2013.

[31] S. B. Shaw and A. K. Singh, "A survey on scheduling and load balancing techniques in cloud computing environment," in *Proc. Int. Conf. Comput. Commun. Technol. (ICCCT)*, Sep. 2014, pp. 87–95.
[32] J. B. Rosen, "Existence and uniqueness of equilibrium points for concave N-person games," *Econ. J. Econ. Soc.*, vol. 33, no. 3, pp. 520–534, 1965.
[33] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, 2008.
[34] N. Kaushik and J. Kumar, "A computation offloading framework to optimize energy utilisation in mobile cloud computing environment," *Int. J. Comput. Appl. Inf. Technol.*, vol. 5, no. 2, pp. 61–69, 2014.
[35] S. Shah and R. Kothari, "Convergence of the dynamic load balancing problem to Nash equilibrium using distributed local interactions," *Inf. Sci.*, vol. 221, pp. 297–305, Feb. 2013.
[36] C. Xavier and S. S. Iyengar, *Introduction to Parallel Algorithms*, vol. 1. Hoboken, NJ, USA: Wiley, 1998.

SANGHONG AHN (S'16) received the B.S. and M.S. degrees from the Korea Advanced Institute of Science and Technology, Daejeon, South Korea, in 2010 and 2012, respectively, where he is currently pursuing the Ph.D. degree with the School of Electronic Engineering. He has contributed several articles to the International Telecommunication Union Telecommunication. His research interests include utilization of Web-based multimedia, the Internet of Things, and cloud computing.

JOOHYUNG LEE (S'09–M'14) received the B.S., M.S., and Ph.D. degrees from the Korea Advanced Institute of Science and Technology, Daejeon, South Korea, in 2008, 2010, and 2014, respectively. From 2012 to 2013, he was a Visiting Researcher with the Information Engineering Group, Department of Electronic Engineering, City University of Hong Kong, Hong Kong. From 2014 to 2017, he was with Samsung Electronics as a Senior Engineer. He is currently an Assistant Professor with the Department of Software, Gachon University, South Korea. He has contributed several articles to the International Telecommunication Union Telecommunication and 3rd Generation Partnership Project. His current research interests include resource allocation and optimization, with a focus on resource management for future media, such as augmented reality and virtual reality, 5G networks, green networks, cloud computing, smart grids (future power grids), and network economics.

Dr. Lee was an Active Member of the GreenTouch Consortium. He was a recipient of Best Paper Award at the Integrated Communications, Navigation, and Surveillance Conference in 2011. He has been a Technical Reviewer for several conferences and journals, such as the IEEE Communications Letters, the IEEE Transactions on Vehicular Technology, and *Computer Communications*.

SANGDON PARK (S'16–M'17) received the B.S., M.S., and Ph.D. degrees from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2011, 2013, and 2017, respectively. He is currently a Post-Doctoral Researcher with the Information and Electronics Research Institute, KAIST. He has contributed several articles to the International Telecommunication Union Telecommunication. His research interests include resource allocation and optimization in smart grids, wireless network, and cloud computing. He received the Best Student Paper Award at the 11th International Conference on Queueing Theory and Network Applications in 2016.

S. H. SHAH NEWAZ (M'13) received the B.Sc. degree in information and communication engineering from East West University, Dhaka, Bangladesh, and the M.Sc. and Ph.D. degrees from the Korea Advanced Institute of Science and Technology (KAIST) in 2010 and 2013, respectively. He served as a Collaborating Researcher at Institut Telecom, Telecom SudParis, France. He served as a Post-Doctoral Researcher at KAIST from 2013 to 2016. He is currently a Lecturer with the School of Computing and Informatics, Universiti Teknologi Brunei, Brunei Darussalam.

His research interests include energy-efficient passive optical networks, optical and wireless converged networks, software-defined networking, mobility and energy efficiency issue in wireless networks, edge cloud/fog computing, smart grid, and content delivery networks, all with specific focus mainly on protocol design and performance aspects.

JUN KYUN CHOI (M'88–SM'00) received the M.S. (Eng.) and Ph.D. degrees in electronic engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 1985 and 1988, respectively.

He was with the Electronics and Telecommunication Research Institute from 1986 to 1997. In 1998, he joined the KAIST as a Professor. His research interests include broadband network architecture and technologies, with particular emphasis on performance and protocol problems.

Prof. Choi has been an Active Member of the International Telecommunication Union Telecommunication Standardization Sector Study Group 13 as a Rapporteur/Editor on the ATM, the MPLS, and the NGN issues since 1993.