Performance Optimization of Serverless Computing for Latency-Guaranteed and Energy-Efficient Task Offloading in Energy-Harvesting Industrial IoT

Haneul Ko, *Member, IEEE*, Sangheon Pack, *Senior Member, IEEE*, and Victor C. M. Leung, *Life Fellow, IEEE*

***Abstract*—Serverless architecture enables various intelligent applications to be run without managing infrastructure. In this architecture, the computing cost is generally proportional to the number of requested stateless functions and this number can affect the task completion time and, thus, it is prominent to decide an appropriate number of requested stateless functions. In this article, we propose a latency-guaranteed and energy-efficient task offloading (LETO) system where an Internet of Things (IoT) device decides the number of stateless functions requested to the cloud by considering the deadline on the task completion time and its energy level. To minimize the computing cost while guaranteeing sufficiently short task completion time and low energy outage probability, we formulate a constrained Markov decision process (CMDP) problem and convert the CMDP problem into an equivalent linear programming (LP) model. By solving the LP model, the optimal policy on the number of requested stateless functions can be achieved. Evaluation results illustrate that LETO can cut down the operating expenditure (OPEX) by up to 59% compared to a latency-guaranteed offloading scheme while keeping the task completion time and the energy outage probability below desirable levels.**

***Index Terms*—Constrained Markov decision process (CMDP), energy harvesting, serverless computing, task offloading.**

I. Introduction

RECENTLY, various intelligent applications requiring huge computing resource (e.g., industry automation, power systems automation, augmented reality (AR), and deep learning) have become more popular in the industrial Internet of Things (IoT) environment [1], [2]. However, IoT devices have limited battery capacities, which is a main implementation barrier of industrial IoT systems [3]. Therefore, industrial and academical researchers have interested on the energy harvesting technique where electricity is derived from external wasted energy [3]–[5]. However, the harvested energy is generally unmanageable and sporadic. Moreover, the energy can be harvested by spatiotemporal characteristics of the energy source.[1] Therefore, it is a challenging problem to supply a sustainable energy to IoT devices. To alleviate this problem, many research have been conducted in [6]–[10]. Among them, the task offloading where an IoT device offloads its tasks for intelligent applications to an external cloud is a promising one.

To support such task offloading, serverless architectures (e.g., Google cloud and AWS Lambda functions) have came out as a novel computation offloading model to improve resource efficiency of the cloud [11]. With these architectures, IoT devices need not to administer dedicated servers or virtual machines to process their tasks. Instead, IoT devices simply define required stateless functions to process their tasks and send request messages on the stateless functions to the cloud.[2] Therefore, there are no transmissions of stateless functions to the cloud.

Meanwhile, many tasks in intelligent applications (e.g., training tasks in deep learning) consist of several subtasks and, thus, can be conducted in a parallel manner. If the requested stateless functions for subtasks can be conducted parallelly,[3] the task completion time can be significantly reduced. However, service operators of serverless architecture (e.g., AWS [12] and Google [13]) charge in proportion to the number of requested stateless functions. Therefore, IoT devices should decide the appropriate number of stateless functions to optimize the tradeoff between the task completion time and the computing cost.

In this article, we propose a latency-guaranteed and energy-efficient task offloading (LETO) system. In LETO, an IoT device decides the number of stateless functions requested to the cloud by considering its energy level and the deadline on the task completion time. To minimize the computing cost while guaranteeing sufficiently short task completion time and low energy outage probability, we formulate a constrained Markov decision process (CMDP) problem and convert the

Manuscript received 31 May 2021; revised 11 October 2021; accepted 17 December 2021. Date of publication 21 December 2021; date of current version 24 January 2023. This work was supported by the National Research Foundation (NRF) of Korea Grant funded by the Korean Government (MSIP) under Grant 2020R1A2C3006786 and Grant 2021R1A4A3022102. *(Corresponding author: Sangheon Pack.)*

Haneul Ko is with the Department of Computer and Information Science, Korea University, Seoul 136-713, South Korea (e-mail: heko@korea.ac.kr).

Sangheon Pack is with the School of Electrical Engineering, Korea University, Seoul 136-713, South Korea (e-mail: shpack@korea.ac.kr).

Victor C. M. Leung is with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China, and also with the Department of Electrical and Computer Engineering, The University of British Columbia, Vancouver, BC V6T 1Z4, Canada (e-mail: vleung@ieee.org).

Digital Object Identifier 10.1109/JIOT.2021.3137291

[1]For example, wind power and/or solar energy intensity change according to the time and location.

[2]The stateless function does not need to keep track of any states for its operation [14]. That is, the stateless function does not take any historical information or outputs from other functions.

[3]Note that IoT devices can easily control the number of functions launched at any point [14]–[16].

2327-4662 © 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: University of Leeds. Downloaded on May 06,2024 at 17:12:00 UTC from IEEE Xplore. Restrictions apply.

CMDP problem into an equivalent linear programming (LP) model. By solving the LP model, the optimal policy on the number of requested stateless functions can be obtained. Evaluation results illustrate that LETO can cut down the operating expenditure (OPEX) by up to 59% compared to a latency-guaranteed offloading scheme while maintaining the task completion time and the energy outage probability below certain levels. Also, it can be found that LETO adjusts its operation according to its operating environments.

The contribution of this article can be summarized as follows: 1) we design a latency-guaranteeing and energy-efficient task offloading system for the serverless computing, while optimizing our system by using the CMDP formulation and 2) comprehensive simulation results are given and scrutinized under diverse environments, which provide useful instructions for designing the task offloading system with a serverless architecture in energy harvesting environments.

The remainder of this article is as follows. Related works are summarized in Section II. LETO is presented in Section III. After that, the CMDP model is formulated in Section IV. The evaluation results are discussed in Section V, and followed by the concluding remarks in Section VI.

II. Related Work

Lots of works have been conducted for the computation offloading in IoT systems [14]–[30]. According to the target environment, these works can be categorized into: 1) device-to-device (D2D)offloading [19]–[23]; 2) general cloud computing [24]–[30]; and 3) serverless computing [14]–[18]. In D2D offloading, no servers are required and each IoT device offloads its task to neighbor IoT devices. In traditional cloud computing (e.g., IaaS and PaaS), dedicated server instances can be allocated in a form of virtual resources in advance, and then IoT device can offload its task to these servers. On the other hand, in serverless computing, the resources to run applications (i.e., stateless function) are provisioned on demand. That is, the allocated resources can be scaled up and down in response to the increased or decreased numbers of requested stateless functions flexibly.

Ko *et al.* [19] introduced a cooperative computation algorithm. In this algorithm, two IoT devices are grouped and they offload an appropriate number of subtasks to the opponent IoT device by taking the elapsed time and their energy levels into consideration. Pu *et al.* [20] suggested a computation offloading framework to diminish the average energy consumption under the incentive constraint. Fan *et al.* [21] formulated a potential game to maximize a social revenue of mobile devices and then proposed an algorithm that decides jointly resource allocation and task offloading. Saleem *et al.* [22] formulated an optimization problem to achieve the minimum task completion time and devised an algorithm that segments the task by considering constraints such as the energy consumption and the task completion deadline. Lin *et al.* [23] decided the ratio of the offloading subtasks for minimizing the weighted energy consumption of mobile devices based on the convex optimization.

In heterogeneous cloud environments, Ko *et al.* [24] designed an offloading algorithm for the spatiotemporal offloading decision by considering the transmission cost, energy consumption, and task completion deadline. Zhou *et al.* [25] introduced a context-aware mobile cloud computing system by considering context changes (e.g., bandwidth, cloud resource, and etc.) for the minimization of energy consumption and the task completion time. Zhao *et al.* [26] designed a dynamic-programming-based algorithm that allocates computational resources and bandwidth to minimize the energy consumption of devices while maintaining the task completion time below the desired level. Wu *et al.* [27] developed a centralized task offloading algorithm to choose reliable edge clouds and allocate subtasks to them. Jin *et al.* [28] formulated an integer LP problem to minimize the cost regarding CPU/GPU, instantiation, and transmission under the constraints of application deadline and resource limitation. Then, they developed a heuristic algorithm having a low complexity to obtain a suboptimal solution. Wu *et al.* [29] formulated a joint optimization problem on whether to offload tasks and how to allocate computing and communication resources to minimize the energy consumption while meeting the task completion deadline. Then, they proposed a heuristic algorithm by exploiting a layered structure of the formulated problem. Under similar objectives, Zhao *et al.* [30] formulated an optimization problem on several decision variables, such as the offloading ratio, transmission power, and computing resource. By decomposing the formulated problem, they designed a heuristic algorithm to decide the offloading ratio, transmission power, and computing resource.

Das *et al.* [15] introduced a performance optimization framework to dynamically choose where to execute stateless functions by estimating the latency and cost for running those functions in the cloud. Wang *et al.* [14] proposed a deep-reinforcement-learning-based scheduler that dynamically controls the number of the stateless functions and their memory size to balance the tradeoff between the quality of the output from the stateless functions and the computing cost. Gupta *et al.* [16] suggested a resource allocation scheme that can achieve the maximized summation of user utilities as a function of the task completion time. Cicconetti *et al.* [17] described how to exploit the ETSI multiaccess edge computing standard to realize the serverless computing. Apostolopoulos *et al.* [18] introduced a flexible resource sharing method in the offloading system where both stateless functions and virtual machines can be exploited for the task processing.

However, these works do not consider the energy depletion problem of IoT devices, which is one of main implementation barriers of IoT systems. If the energy of IoT devices is depleted, the system operator replace their batteries, which can lead high OPEX.

III. LETO System

The system model of this article is shown in Fig. 1. The application runs on the IoT device and generates periodically a

Authorized licensed use limited to: University of Leeds. Downloaded on May 06,2024 at 17:12:00 UTC from IEEE Xplore. Restrictions apply.

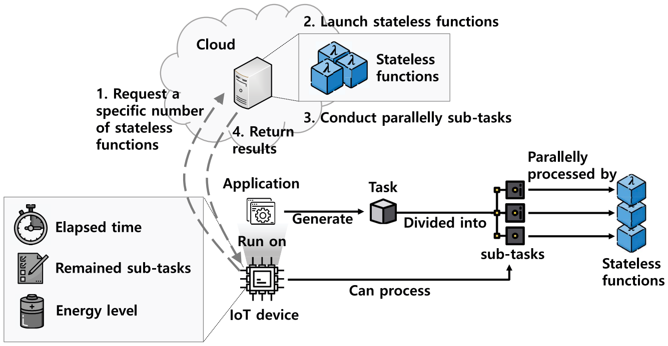


Fig. 1. System model.

task with the rate λ_T. The task can be partitioned into K types of subtasks, and they can be conducted in a parallel manner (e.g., training tasks in deep learning). For example, Tensorflow supports a distributed training method where multiple workers train over different slices of input data parallelly [31]. As another example, AWS Lambda provides several stateless functions such as the data preprocessing function that preprocesses data before feeding it to deep learning models [32]. This preprocessing does not need any historical information or outputs from other functions and, thus, preprocessing functions can operate in a parallel manner.

Meanwhile, the IoT device has the energy harvesting capability. Generally, the IoT device can harvest the energy only when the energy source provides a sufficient energy. For example, only when the wind strength is above a certain level, the IoT device having wind-based energy harvesting capability can harvest the energy. Therefore, a Bernoulli random process can be used to describe whether to harvest the energy or not (i.e., one unit energy can be harvested with the harvesting probability p_H [35]). By consuming the harvested energy, the IoT device can process the kth type subtasks with the processing rate $\mu_{I,k}$. However, the energy of the IoT device can be depleted when it processes all subtasks by itself. In addition, the IoT device cannot process the whole task within a sufficiently short task completion time due to its limited computing power. Therefore, we consider a serverless architecture where the cloud can launch several stateless functions for processing subtasks offloaded from an IoT device.[4] Note that the input data (e.g., training set in deep learning) can be maintained at the cloud and each subtask (e.g., training) can be processed by a single stateless function. The operational procedure of the serverless architecture is as follows. First, the IoT device requests a specific number A_k (for $\forall k$) of stateless functions, which can process the kth type subtasks (for $\forall k$), to the cloud (step 1 in Fig. 1). After receiving the request message, the cloud launches several stateless functions for the kth type subtasks according to the requested number A_k (step 2 in Fig. 1). Then, the stateless functions conduct parallelly subtasks (i.e., each stateless function processes a subtask in a parallel manner) (step 3 in Fig. 1), which indicates that the task completion time can be significantly reduced. After finishing the subtask at each stateless function, the cloud returns the results of the subtasks to the IoT device (step 4 in Fig. 1).

Because stateless functions can operate in a parallel manner [14], the more stateless functions are employed, the shorter task completion time can be obtained. However, the commercial serverless architecture (i.e., AWS and Google) generally charges in proportion to the total number of requested stateless functions, $\sum_k A_k$. That is, if the unit cost to use one stateless function is ρ_S and $\sum_k A_k$ stateless functions are requested, the computing cost is given by $\rho_S \sum_k A_k$. To sum up, there is a tradeoff between the task completion time and the computing cost. To balance this tradeoff, the IoT device decides: 1) whether to request stateless functions or not and 2) the appropriate number of requested stateless functions by considering the elapsed time from the task generation, remained subtasks, and energy level. These two kinds of decisions can be handled by one dimension of action space, i.e., the number of requested stateless functions, because the action representing zero requested stateless function can be interpreted as the situation where the IoT device decides not to request any stateless function.

We can consider the following operational examples of the IoT device. As a first example, when it is assumed that the elapsed time from the task generation is short and the IoT device has lots of energy, the IoT device decides not to request any stateless function and conducts the task computing in a local manner to reduce the computing cost. On the other hand, if the elapsed time nearly reaches to the task completion deadline, the IoT devices requests several stateless functions for faster processing. For the optimal decision of the IoT device, the CMDP problem is formulated in the next section.

IV. Constraint Markov Decision Process (CMDP)

In the CMDP model, the agent conducts successively a specific action to minimize (or maximize) the cost (or reward) under the defined constraints [33]. That is, in this article, the IoT device decides whether to request stateless functions or not and appropriate number of requested stateless functions at the time epochs, $T = \{1, 2, 3, \ldots\}$, to minimize the computing cost while satisfying the constraints on the task completion time and the energy outage probability.

A. State Space

The state space $\mathbf{S}$ is defined as

$$\mathbf{S} = \mathbf{E} \times \mathbf{L} \times \prod_k \mathbf{R_k} \tag{1}$$

where $\mathbf{E}$ denotes the energy level of the IoT device. $\mathbf{L}$ and $\mathbf{R_k}$ represent the elapsed time from the task generation and the number of remained kth type subtasks (that are not completed yet), respectively. Also, $\times$ represents a Kronecker product.

$\mathbf{E}$ can be represented as

$$\mathbf{E} = \{0, 1, 2, \ldots, E_{\max}\} \tag{2}$$

where $E_{\max}$ is the maximum battery capacity of the IoT device.

[4]It is assumed that a stateless function has higher processing speed than the IoT device. Specifically, the processing rate of a stateless function for the kth type subtasks is $\mu_{C,k}$ ($> \mu_{I,k}$).

Authorized licensed use limited to: University of Leeds. Downloaded on May 06,2024 at 17:12:00 UTC from IEEE Xplore. Restrictions apply.

Meanwhile, when $R_{\max,k}$ represents the total number of the kth type subtasks that constitute the task, $\mathbf{R_k}$ can be expressed as

$$\mathbf{R_k} = \{0, 1, 2, \dots, R_{\max,k}\}. \tag{3}$$

$\mathbf{L}$ can be described by

$$\mathbf{L} = \{0, 1, 2, \dots, L_{\max}\} \tag{4}$$

where $L_{\max}$ is the maximum elapsed time.

B. Action Space

The IoT device can request stateless functions for K types of subtasks. Thus, we can define the action space $\mathbf{A}$ by

$$\mathbf{A} = \prod_k \mathbf{A_k} \tag{5}$$

where $\mathbf{A_k}$ represents the action space for the kth type subtasks, which can be described by

$$\mathbf{A_k} = \{0, 1, 2, \dots, N_{\max}\} \tag{6}$$

where $N_{\max,k}$ denotes the maximum number of requested stateless functions for the kth type subtasks. A_k represents the number of stateless functions for the kth type subtasks. Note that $A_k = 0$ denotes the situation where the IoT device decides not to request any stateless function for the kth type subtasks and conducts the computing for the kth type subtasks in a local manner.

C. Transition Probability

If the IoT device decides to process some subtasks by itself, it consumes the energy. Therefore, the next state of the energy level is altered by the chosen action A. Meanwhile, several stateless functions can operate in a parallel manner [14]. Thus, if subtasks are evenly distributed to each stateless function, the processing speed increases proportionally to the number of requested stateless functions. Thus, the transition of the state for the number of remained kth type subtasks $\mathbf{R_k}$ is altered by the chosen action for the kth type subtasks, A_k. If the IoT device does not have enough energy, it cannot conduct the processing for any subtask by itself. Therefore, the transition of $\mathbf{R_k}$ is influenced by the energy level E as well. In addition, we assume that a single task generates at the same time and, therefore, the task can generate only when there is no remained any subtask. That is, the transition of $\mathbf{R_k}$ is affected by the state for the number of remained other types of subtasks. Because the elapsed time denotes the time from the task generation and this time should be reset when completing the task, the transition of the state for the elapsed time $\mathbf{L}$ is influenced by the state of the remained kth type subtasks $\mathbf{R_k}$. Meanwhile, the other states transit with each other in a independent manner. As a result, the transition probability from $S = [E, L, R_1, R_2, \dots, R_K]$ to $S' = [E', L', R'_1, R'_2, \dots, R'_K]$ can be represented by (7), shown at the bottom of the page.

The energy consumption of requesting the stateless functions is much smaller than that of processing the subtasks because we assume the task with high complexity. Therefore, it can be assumed that the IoT device does not consume the energy for requesting the stateless functions to the cloud. In addition, since the input data (e.g., training set in deep learning) is already maintained at the cloud, there is no energy consumption to transmit the input data. Meanwhile, the IoT device can harvest one unit energy if a condition of the energy source is satisfied (with the harvesting probability p_H [35]).[5] Note that this harvesting probability can be different according to the energy conversion efficiency, location of the IoT device, and so on.

Meanwhile, the IoT device consumes one unit energy when processing the kth type subtask by itself (i.e., $A_k = 0$).[6] Thus, the total energy consumption ε_T for the processing subtasks in the IoT device can be calculated as $\sum_k \delta[A_k = 0]$, where $\delta[\cdot]$ is a function that returns one if a given condition (e.g., $A_k = 0$) is true; otherwise, it returns zero. The IoT device cannot process any subtasks when it does not have enough energy. In this situation, there is no energy consumption in the IoT device. Therefore, the related transition probabilities can be described as (8) and (9), shown at the bottom of the page.

If the IoT device decides not to conduct any processing of subtasks (i.e., $\sum_k \delta[A_k \neq 0] = K$), it consumes no energy. Therefore, the IoT device can harvest one unit of energy when its battery has a room (i.e., $E \neq E_{\max}$). Thus, the correlated

[5]IoT devices have different energy harvesting probabilities and, thus, their energy levels change differently with each other. In addition, an IoT device without any energy harvesting capability can be considered by setting the harvesting probability p_H as 0.

[6]Since general IoT devices do not have any CPU processing rate adaptation functionality, a constant CPU processing rate for IoT devices can be assumed. In addition, the energy consumption of the IoT device in processing a subtask during the decision epoch can be matched to the one unit energy.

$$P\big[S'|S, A\big] = P\big[E'|E, A\big] \times P\big[L'|L, R_1, R_2, \dots, R_K\big] \times \prod_k P\big[R'_k|R_1, \dots, R_k, ..R_K, E, A_k\big] \tag{7}$$

$$P\left[E'|E \geq \varepsilon_T, \sum_k \delta[A_k = 0] > 0\right] = \begin{cases} p_H, & \text{if } E' = E \\ 1 - p_H, & \text{if } E' = E - \varepsilon_T \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

$$P\left[E'|E < \varepsilon_T, \sum_k \delta[A_k = 0] > 0\right] = \begin{cases} p_H, & \text{if } E' = E + 1 \\ 1 - p_H, & \text{if } E' = E \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

Authorized licensed use limited to: University of Leeds. Downloaded on May 06,2024 at 17:12:00 UTC from IEEE Xplore. Restrictions apply.

transition probabilities can be denoted as

$$P\left[E'|E, \sum_k \delta\left[A_k \neq 0\right] = K\right] = \begin{cases} p_H, & \text{if } E' = E+1 \\ 1-p_H, & \text{if } E' = E \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

and

$$P\left[E'|E = E_{\max}, \sum_k \delta\left[A_k \neq 0\right] = K\right] = \begin{cases} 1, & \text{if } E' = E \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

It is assumed that the time duration between successive task occurrences at the IoT device follows an exponential distribution, and its average is $1/\lambda_T$. With this assumption, the task occurrence probability within the decision epoch duration τ can be calculated as $\lambda_T\tau$ [9]. Right after the task occurs, the number of kth type subtasks becomes $R_{\max,k}$. Meanwhile, because we assume that a single task generates at the same time, the task can generate only when any subtask does not remain (i.e., $R_k = 0$ for $\forall k$). Thus, the related transition probability can be represented as (12), shown at the bottom of the page.

The processing time of a single stateless function for the kth type subtask is assumed to follow an exponential distribution with mean $1/\mu_{C,k}$. Then, the probability that a single stateless function completes a subtask during τ can be obtained as $\mu_{C,k}\tau$ [9]. Also, the probability is proportional to the number of requested stateless functions A_k. Accordingly, the corresponding transition probability can be represented as (13), shown at the bottom of the page.

The processing time of the kth type subtask in the IoT device is assumed to follow an exponential distribution with mean $1/\mu_{I,k}$. Thus, the probability that the IoT device completes a kth type subtask during the decision epoch can be calculated as $\mu_{I,k}\tau$. Meanwhile, the IoT device can process subtasks only when it has enough energy (i.e., $E \geq \varepsilon_T$). Accordingly, the related transition probabilities can be denoted as (14) and (15), shown at the bottom of the page.

When there is no remained subtask, the elapsed time is always 0. Thus, $P[L'|L, R_1 = 0, R_2 = 0, \ldots, R_K = 0]$ can be represented by

$$P\left[L'|L, R_1 = 0, R_2 = 0, \ldots, R_K = 0\right] = \begin{cases} 1, & \text{if } L' = 0 \\ 0, & \text{otherwise.} \end{cases} \tag{16}$$

On the other hand, when there are remained any subtasks (i.e., $\sum_k \delta[R_k \neq 0] \neq 0$), the elapsed time increases one by one until it reaches the maximum time. Therefore, $P[R'_k|R_1, \ldots, R_k \neq 0, \ldots, R_K, E \geq \varepsilon_T, A_k = 0]$ and $P[L'|L = L_{\max}, \sum_k \delta[R_k \neq 0] \neq 0]$ can be represented by

$$P\left[L'|L \neq L_{\max}, \sum_k \delta\left[R_k \neq 0\right] \neq 0\right] = \begin{cases} 1, & \text{if } L' = L+1 \\ 0, & \text{otherwise} \end{cases} \tag{17}$$

and

$$P\left[L'|L = L_{\max}, \sum_k \delta\left[R_k \neq 0\right] \neq 0\right] = \begin{cases} 1, & \text{if } L' = L_{\max} \\ 0, & \text{otherwise.} \end{cases} \tag{18}$$

D. Cost & Constraint Functions

1) Cost Function: To minimize the computing cost, the cost function $r(S, A)$ is defined. The computing cost is proportional to the number of requested stateless functions $\sum_k A_k$. Hence, $r(S, A)$ can be defined as

$$r(S, A) = \rho_S \sum_k A_k \tag{19}$$

where ρ_S denotes the unit cost to use one stateless function during the decision epoch.

2) Constraint Functions: Right after the number of all subtasks becomes 0, the elapsed time L denotes the task completion time. Note that, in our system, IoT devices just transmit small size of request messages on how many stateless functions are launched in the cloud. That is, the transmission latency for this request message is much smaller than the processing latency of subtasks and, therefore, the transmission latency for the request message can be neglected in computing the task completion time. Therefore, the constraint function

$$P\left[R'_k|R_1 = 0, \ldots, R_k = 0, \ldots, R_K = 0, E, A_k\right] = \begin{cases} \lambda_T\tau, & \text{if } R' = R_{\max,k} \\ 1-\lambda_T\tau, & \text{if } R' = 0 \\ 0, & \text{otherwise} \end{cases} \tag{12}$$

$$P\left[R'_k|R_1, \ldots, R_k \neq 0, \ldots, R_K, E, A \neq 0\right] = \begin{cases} A_k\mu_{C,k}\tau, & \text{if } R'_k = R_k - 1 \\ 1-A\mu_{C,k}\tau, & \text{if } R'_k = R_k \\ 0, & \text{otherwise} \end{cases} \tag{13}$$

$$P\left[R'_k|R_1, \ldots, R_k \neq 0, \ldots, R_K, E \geq \varepsilon_T, A_k = 0\right] = \begin{cases} \mu_{I,k}\tau, & \text{if } R'_k = R_k - 1 \\ 1-\mu_{I,k}\tau, & \text{if } R'_k = R_k \\ 0, & \text{otherwise} \end{cases} \tag{14}$$

$$P\left[R'_k|R_1, \ldots, R_k \neq 0, \ldots, R_K, E < \varepsilon_T, A_k = 0\right] = \begin{cases} 1, & \text{if } R'_k = R_k \\ 0, & \text{otherwise} \end{cases} \tag{15}$$

Authorized licensed use limited to: University of Leeds. Downloaded on May 06,2024 at 17:12:00 UTC from IEEE Xplore. Restrictions apply.

$c_L(S, A)$ for the task completion time can be represented by

$$c_L(S, A) = \prod_k \delta[R_k = 0]L. \quad (20)$$

We define the constraint function $c_E(S, A)$ for the energy outage probability. Energy outage means a situation where any energy does not remain in the battery of the IoT device (i.e., $E = 0$). Thus, we can define $c_E(S, A)$ as

$$c_E(S, A) = \delta[E = 0]. \quad (21)$$

E. Optimization Formulation

The average computing cost ζ_C can be defined as

$$\zeta_C = \lim_{t\to\infty} \sup \frac{1}{t} \sum_{t'}^{t} E[r(S_{t'}, A_{t'})] \quad (22)$$

where S_t and A_t denote the state and the chosen action at $t \in T$, respectively.

Meanwhile, the average task completion time can be represented by

$$\psi_L = \lim_{t\to\infty} \sup \frac{1}{t} \sum_{t'}^{t} E[c_L(S_{t'}, A_{t'})]. \quad (23)$$

In addition, the average energy outage probability can be represented as

$$\psi_E = \lim_{t\to\infty} \sup \frac{1}{t} \sum_{t'}^{t} E[c_E(S_{t'}, A_{t'})]. \quad (24)$$

We can express the CMDP model as

$$\min_{\pi} \ \zeta_C \quad (25)$$

$$\textbf{s.t.} \ \ \psi_L \leq \theta_L \text{ and } \psi_E \leq \theta_E \quad (26)$$

where π is a policy that implies the probabilities of choosing a specific action at each state. θ_L and θ_E denote the upper limits on the average task completion time and the average energy outage probability, respectively.

The formulated CMDP model can be transformed to an equivalent LP model by defining the stationary probabilities of state S and action A, $\varphi(S, A)$, as decision variables of the LP model. The LP model can be represented by

$$\min_{\varphi(S,A)} \ \sum_S \sum_A \varphi(S, A) r(S, A) \quad (27)$$

$$\text{subject to} \ \sum_S \sum_A \varphi(S, A) c_L(S, A) \leq \theta_L \quad (28)$$

$$\sum_S \sum_A \varphi(S, A) c_E(S, A) \leq \theta_E \quad (29)$$

$$\sum_A \varphi\left(S', A\right) = \sum_S \sum_A \varphi(S, A) P\left[S' | S, A\right] \quad (30)$$

$$\sum_S \sum_A \varphi(S, A) = 1 \quad (31)$$

and

$$\varphi(S, A) \geq 0. \quad (32)$$

The equation in (27) shows the objective function to minimize the average computing cost. The constraints in (28) and (29) denote the same constraints of the CMDP model in (26). In addition, the constraint in (30) is for the Chapman–Kolmogorov equation. The constraints in (31) and (32) are needed for keeping the probability properties.

By solving the LP problem, we can obtain the optimal stochastic policy $\pi^*(S, A)$ as the solution of the CMDP model. The optimal stochastic policy means that an action A at state S is selected based on the optimal probability distribution. If there is no solution to satisfy all constraints, the IoT device does not request any stateless function. When a policy table representing the probabilities of choosing action A at each state S is constructed by the cloud and delivered to the IoT device, it can operate with the optimal policy without high computational overhead.

V. Evaluation Results

To evaluate the performance of LETO, we consider the following four comparison schemes: 1) LATENCY [15] where an IoT device minimizes the computing cost while maintaining the task completion time below the desired level; 2) IoT where an IoT device processes all subtasks by itself; 3) MAX where an IoT device requests always the maximum number of stateless functions; and 4) RAND where an IoT device decides randomly its action. The objective of this article is to minimize the average computing cost ζ_C while maintaining the average task completion time ψ_L and the average outage probability ψ_E below their upper limits. Therefore, ζ_C, ψ_L, and ψ_E are used as the performance measures. In addition, the average OPEX ζ_O of the IoT device, which consists of the average computing cost and the average battery replacement cost,[7] is used as another performance measure.

The default parameter settings are as follows. Because the computing capacity of an IoT device is lower than that of the cloud, the processing speeds of a single stateless function and an IoT device, denoted by μ_C and μ_I, are set to 0.2 (1/s) and 0.1 (1/s), respectively. Meanwhile, since human resources are needed for the battery replacement and human resources are generally high priced, it is assumed that the cost ρ_R to replace the battery is more expensive than the unit cost ρ_S to use one stateless function during the decision epoch. Therefore, ρ_S and ρ_R are set to 1 and 3, respectively.[8] The energy harvesting probability p_H is 0.2 [35]. The number of subtask types is 2 [36]. The maximum batter capacity, $E_{\max}$, and the total number of subtasks that constitute the task, $R_{\max}$, are set to 5 and 4, respectively. In addition, the maximum elapsed time $L_{\max}$ and the maximum number of requested stateless functions, $N_{\max}$ are set to 10 and 3, respectively. The upper limits on the average task completion time and the average energy outage probability are 3.5 (s) and 0.01 [37], respectively. Meanwhile,

[7]It is assumed that the battery of IoT device can be replaced as a new one if it is depleted.

[8]The effect of the ratio between these two costs will be described in Section V-F.

Authorized licensed use limited to: University of Leeds. Downloaded on May 06,2024 at 17:12:00 UTC from IEEE Xplore. Restrictions apply.

TABLE I
DEFAULT PARAMETER SETTINGS

Parameter	$E_{\max}$	$R_{\max}$	$L_{\max}$	$N_{\max}$	p_H	θ_L (sec)	θ_E	λ_T (1/sec)	μ_C (1/sec)	μ_I (1/sec)	ρ_S	ρ_R
Value	5	4	10	3	0.2	3.5	0.01	0.7	0.2	0.1	1	3

TABLE II
RUNNING TIME (UNIT: SEC) TO SOLVE THE LP MODEL

K	1	2	3	4	5
Running time	0.21	3.36	13.36	54.76	215.36

the task occurrence rate λ_T is 0.7 (1/s) [38]. These parameter settings are summarized in Table I.[9]

A. Running Time to Solve the LP Model

Table II shows the running time to solve the LP model. We have exploited Intel i9-10900K CPU, 64G RAM, and an LP solver of MATLAB. As shown in Table II, it can be found that the running time increases as the number of subtask types, K, increases. This is because the complexity of the LP model is a polynomial function of the solution space [39] and the solution space in our LP model is proportional to the number of subtask types. Therefore, even with five subtask types, the running time is not quite high. In addition, the IoT device can store the optimal policy on the number of requested stateless functions in a table form. Then, the IoT device decides the number of requested stateless functions by following the policy in the table. This indicates that the IoT device needs not to consume the time to solve the LP model whenever offloading. As a result, the proposed system can be implemented in a practical manner.

B. Effect of θ_L

Fig. 2 illustrates the effect of the upper limit of the average task completion time, θ_L, on the average OPEX ζ_O, the average task completion time ψ_L, and the average energy outage probability ψ_E. As shown in Fig. 2, LETO can minimize the average OPEX while maintaining the average task completion time and the average energy outage probability below desired levels. This can be explained as follows. In LETO, an IoT device requests an appropriate number of stateless functions to the cloud by considering the elapsed time. For example, if the elapsed time is close to the upper limit of the average task completion time, the IoT device requests an increased number of stateless functions to process the remaining subtasks rapidly. Otherwise, the IoT device tries to minimize the number of requested stateless functions to reduce the computing cost. In addition, to maintain the energy outage probability below its upper limit and to avoid any energy depletion, the IoT device does not process any subtasks by itself if it does not have sufficient energy . As a result, the cost to replace the battery of the IoT device can be minimized.

[9] Unfortunately, we were not able to find any specific references about some system-dependent parameters and, thus, we have conducted extensive evaluations with various settings for those parameters. However, we could not find any specific tendency according to those parameters. Therefore, we have included only the results using the default parameter setting.

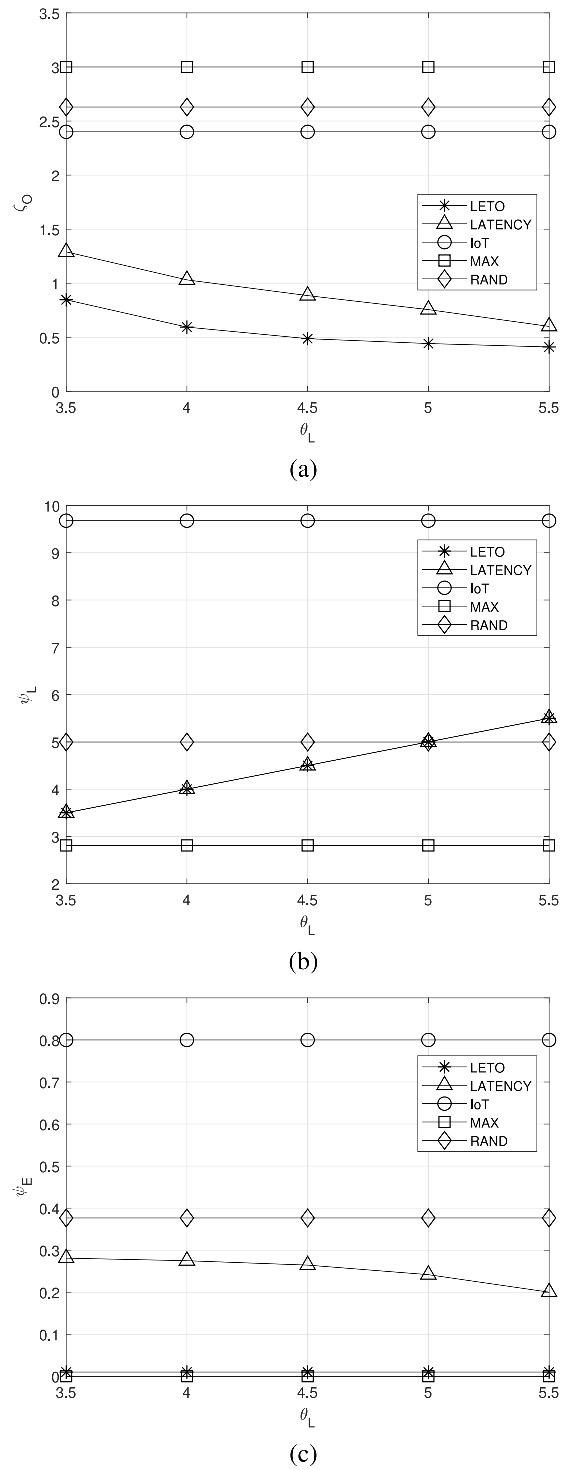


Fig. 2. Effect of θ_L. (a) Average OPEX. (b) Average task completion time. (c) Average energy outage probability.

From Fig. 2(a) and (b), it can be found that LETO operates adaptively according to the upper limit θ_L of the average task completion time. That is, as the upper limit θ_L increases, the

Authorized licensed use limited to: University of Leeds. Downloaded on May 06,2024 at 17:12:00 UTC from IEEE Xplore. Restrictions apply.

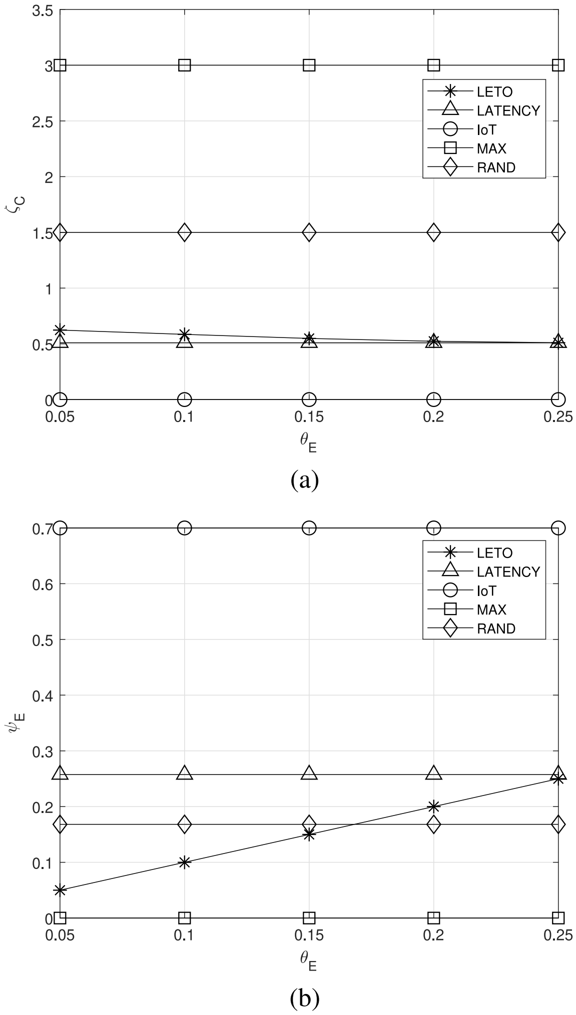


Fig. 3. Effect of θ_E. (a) Average computing cost. (b) Average energy outage probability.

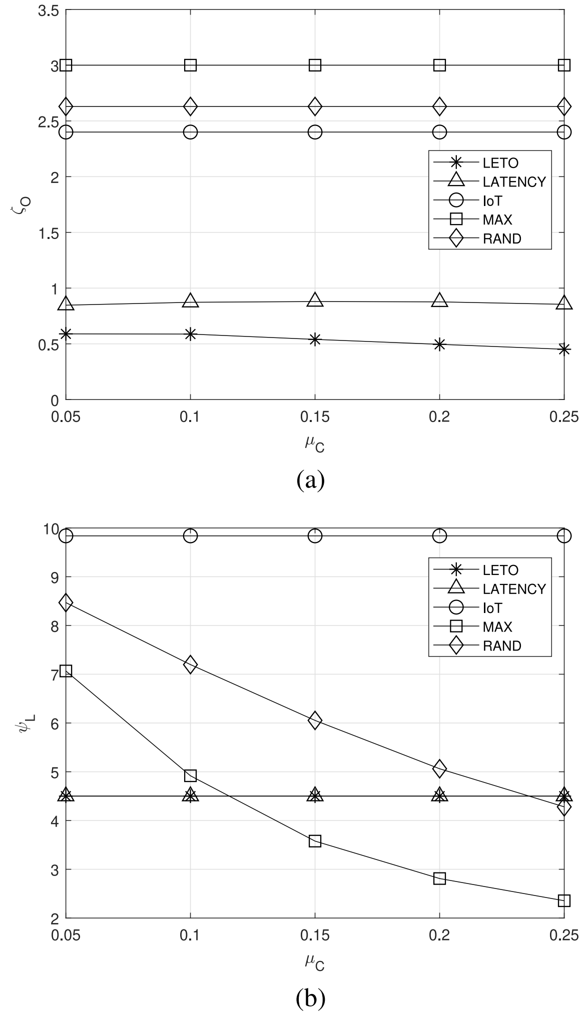


Fig. 4. Effect of μ_C. (a) Average OPEX (b) Average task completion time.

IoT device processes more subtasks by itself, which causes the increased task completion time [see Fig. 2(b)] but reduces the computing cost constituting OPEX [see Fig. 2(a)].

C. Effect of θ_E

Fig. 3(a) and (b) shows the effect of the upper limit of the energy outage probability, θ_E, on the average computing cost and the average energy outage probability, respectively. As shown in Fig. 3(a) and (b), it can be observed that, as θ_E increases, the average computing cost of LETO decreases at the expense of the energy outage probability. This is because a large θ_E means that the IoT device can process more subtasks by itself at the risk of the energy outage (i.e., IoT device can reduce the number of requested stateless functions). On the other hand, the average computing cost and the average energy outage probability of other schemes do not change regardless of θ_E, because they operate without considering any energy outage probability.

D. Effect of μ_C

The effect of the processing speed of a stateless function μ_C is illustrated in Fig. 4. From Fig. 4(b), it can be found that the average task completion times of MAX and RAND dramatically decrease with the increase of μ_C. This is because these schemes use more stateless functions to process subtasks. Also, it can be found that the average task completion time of LETO is constant regardless of μ_C. However, as μ_C increases, LETO can reduce OPEX as shown in Fig. 4(a). This can be explained as follows. When the processing speed of a stateless function increases, the task can be completed within the upper limit of the task completion time even with a less number of stateless functions. In this situation, LETO requests as less stateless functions as possible to complete the task within the upper limit.

E. Effect of p_H

Fig. 5 demonstrates the effect of the energy harvesting probability p_H. From Fig. 5(b) and (c), it can be found that LETO maintains the average task completion time and the average energy outage probability below certain levels regardless of changing p_H. Meanwhile, from Fig. 5(a), it can be shown that the average computing cost of LETO decreases with the increase of p_H. This is because a higher p_H means that the energy may not be depleted even when the IoT

Authorized licensed use limited to: University of Leeds. Downloaded on May 06,2024 at 17:12:00 UTC from IEEE Xplore. Restrictions apply.

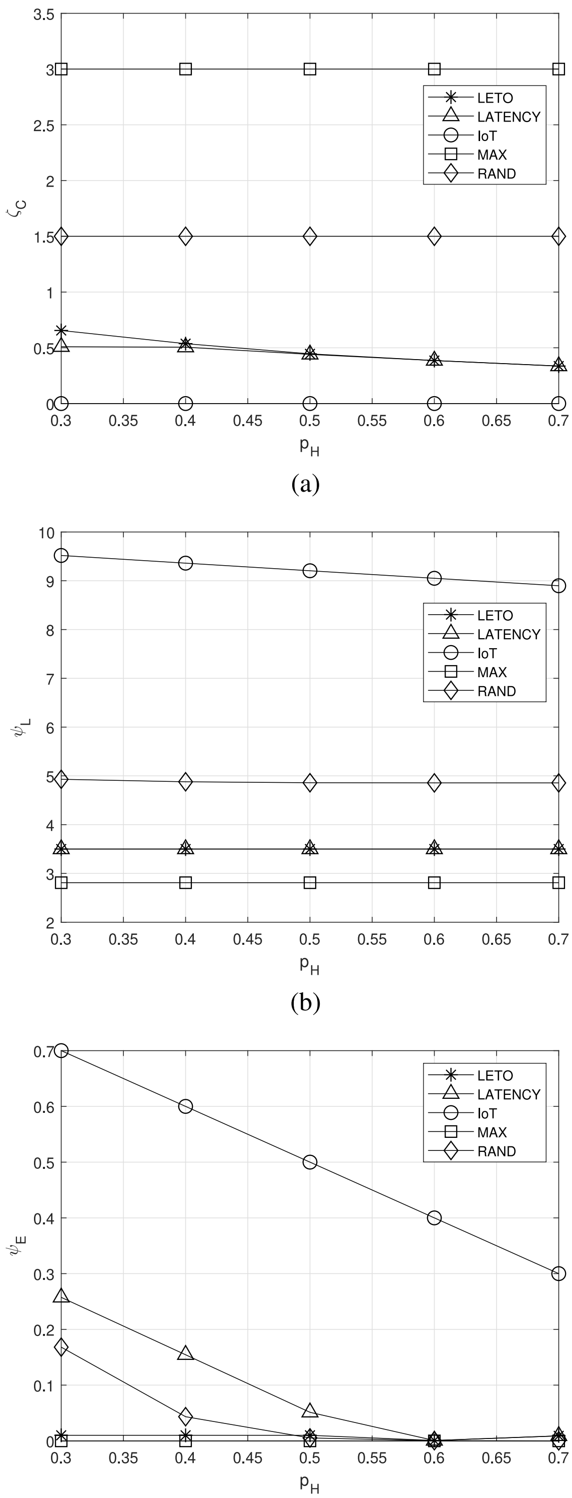


Fig. 5. Effect of p_H. (a) Average computing cost. (b) Average task completion time. (c) Average energy outage probability.

device processes its subtasks aggressively.[10] In LETO, the IoT device recognizes this situation and processes more subtasks adaptively and, thus, the computing cost can be reduced.

[10]In this context, the average energy outage probability of comparison schemes (i.e., LATENCY, IoT, and RAND) significantly decreases with the increase of p_H.

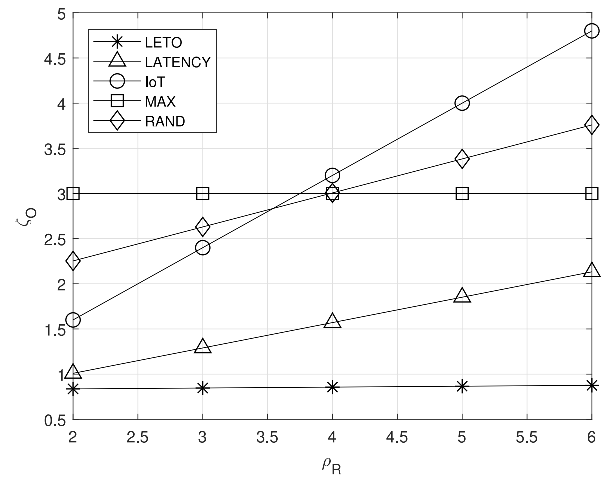


Fig. 6. Effect of ρ_R on the average OPEX.

TABLE III
EFFECT OF $N_{\max}$ ON THE AVERAGE OPEX

Scheme \ $N_{\max}$	4	5	6	7	8
LETO	0.8643	0.8585	0.8563	0.8562	0.8561
LATENCY	1.1788	1.1850	1.1559	1.1264	1.1081
IoT	2.4000	2.4000	2.4000	2.4000	2.4000
MAX	3.0000	4.0000	5.0000	6.0000	7.0000
RAND	2.6296	2.8309	3.0857	3.3935	3.7518

Meanwhile, other comparison schemes except LATENCY do not change their policies regardless of p_H and, therefore, their costs of the serverless computing are constant. In LATENCY, there are few cases where the IoT device cannot process its task due to its energy depletion when p_H is high, which indicates that there is no case where the IoT device in LATENCY requests stateless functions due to the concern about its energy depletion. Therefore, its computing cost slightly decreases with the increase of p_H.

F. Effect of ρ_R

Fig. 6 shows the effect of the cost ρ_R to replace the battery of the IoT device on the average OPEX. From Fig. 6, it can be found that the average OPEX of the comparison schemes except MAX increases as ρ_R increases. This is because a large ρ_R means that high cost occurs when the energy is depleted. On the other hand, the average OPEX of LETO is almost constant regardless of ρ_R. This is because the IoT device in LETO has very low energy outage probability (i.e., 0.01). In this situation, the battery of the IoT device does not need to be changed frequently. Meanwhile, the IoT device in MAX does not consume any energy to process its task and, thus, the average OPEX of MAX is constant regardless of ρ_R.

G. Effect of N_{max}

The effect of $N_{\max}$ on the average OPEX is shown in Table III. Interestingly, the average OPEX of LETO and LATENCY decreases as the maximum number of requested stateless functions increases. This is because a large maximum

Authorized licensed use limited to: University of Leeds. Downloaded on May 06,2024 at 17:12:00 UTC from IEEE Xplore. Restrictions apply.

number of requested stateless functions means high possibility that the tasks can be completed within a short duration. Therefore, for a large $N_{\max}$, the IoT device in LETO and LATENCY can try to reduce the number of requested stateless functions even when the task completion time is close to its upper limit.

VI. Conclusion

In this article, we introduced the LETO system where an IoT device decides whether to request stateless function or not and the number of stateless functions by considering its energy level and the deadline of the task completion time. To optimize the performance of LETO, a CMDP problem was formulated and a practical solution has been derived from the equivalent LP model. Evaluation results demonstrated that LETO requests an appropriate number of stateless functions to the cloud according to the situation (e.g., elapsed time, remained subtasks, etc.) to minimize OPEX while obtaining a sufficiently short task completion time and low energy outage probability. In addition, it can be found that LETO operates adaptively by considering its operating environments, such as the energy harvesting probability, the processing speed of a stateless function, and the maximum number of requested stateless functions. In our future work, we will investigate how to extend LETO for edge-central cloud collaborative environments. In addition, we will investigate a deep-reinforcement-learning-based approach to solve our formulated problem.

References

[1] W. Zhang, B. Han, and P. Hui, "On the networking challenges of mobile augmented reality," in *Proc. ACM SIGCOMM Workshop VR/AR Netw.*, Aug. 2017, pp. 24–29.

[2] H. Ko and S. Pack, "Distributed device-to-device offloading system: Design and performance optimization," *IEEE Trans. Mobile Comput.*, vol. 20, no. 10, pp. 2949–2960, Oct. 2021.

[3] H. Yetgin, K. T. K. Cheung, M. El-Hajjar, and L. H. Hanzo, "A survey of network lifetime maximization techniques in wireless sensor networks," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 2, pp. 828–854, 2nd Quart., 2017.

[4] U. Saleem, S. Jangsher, H. K. Qureshi, and S. A. Hassan, "Joint subcarrier and power allocation in the energy-harvesting-aided D2D communication," *IEEE Trans. Ind. Informat.*, vol. 14, no. 6, pp. 2608–2617, Jun. 2018.

[5] J. Wang, C. Jiang, H. Zhang, Y. Re, K.-C. Chen, and L. Hanzo, "Thirty years of machine learning: The road to Pareto-optimal wireless networks," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 1472–1514, 3rd Quart., 2020.

[6] R. Du, L. Gkatzikis, C. Fischione, and M. Xiao, "On maximizing sensor network lifetime by energy balancing," *IEEE Trans. Control Netw. Syst.*, vol. 5, no. 3, pp. 1206–1218, Sep. 2018.

[7] S. N. Das, S. Misra, B. E. Wolfinger, and M. S. Obaidat, "Temporal-correlation-aware dynamic self-management of wireless sensor networks," *IEEE Trans. Ind. Informat.*, vol. 12, no. 6, pp. 2127–2138, Dec. 2016.

[8] D. Niyato, D. I. Kim, P. Wang, and L. Song, "A novel caching mechanism for Internet of Things (IoT) sensing service with energy harvesting," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2016, pp. 1–6.

[9] H. Ko, S. Pack, and V. C. M. Leung, "Spatiotemporal correlation-based environmental monitoring system in energy harvesting Internet of Things (IoT)," *IEEE Trans. Ind. Informat.*, vol. 15, no. 5, pp. 2958–2968, May 2019.

[10] M. S. Hossain, C. I. Nwakanma, J. M. Lee, and D.-S. Kim, "Edge computational task offloading scheme using reinforcement learning for IIoT scenario," *ICT Exp.*, vol. 6, no. 4, pp. 291–299, Dec. 2020.

[11] S. Hendrickson *et al.*, "Serverless computation with OpenLambda," *8th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 16)*. Denver, CO, USA: USENIX Assoc., Jun. 2016.

[12] Serverless on AWS—Build and Run Applications Without Thinking About Servers." AWS. [Online]. Available: https://aws.amazon.com/serverless (Accessed: Oct. 7, 2021).

[13] "Serverless Computing." Google. [Online]. Available: https://cloud.google.com/serverless (Accessed: Oct. 7, 2021).

[14] H. Wang, D. Niu, and B. Li, "Distributed machine learning with a serverless architecture," in *Proc. IEEE INFOCOM*, Paris, France, May 2019, pp. 1288–1296.

[15] A. Das, S. Imai, S. Patterson, and M. P. Wittie, "Performance optimization for edge-cloud serverless platforms via dynamic task placement," in *Proc. IEEE/ACM CCGRID*, May 2020, pp. 41–50.

[16] V. Gupta, S. R. Phade, T. Courtade, and K. Ramchandran, "Utility-based resource allocation and pricing for serverless computing," Aug. 2020, *arXiv:2008.07793*.

[17] C. Cicconetti, M. Conti, A. Passarella, and D. Sabella, "Toward distributed computing environments with serverless solutions in edge systems," *IEEE Commun. Mag.*, vol. 58, no. 3, pp. 40–46, Mar. 2020.

[18] P. A. Apostolopoulos, E. E. Tsiropoulou, and S. Papavassiliou, "Risk-aware social cloud computing based on serverless computing model," in *Proc. IEEE GLOBECOM*, Dec. 2019, pp. 1–6.

[19] H. Ko, J. Lee, S. Jang, J. Kim, and S. Pack, "Energy efficient cooperative computation algorithm in energy harvesting Internet of Things," *Energies*, vol. 12, no. 21, p. 4050, Dec. 2019.

[20] L. Pu, X. Chen, J. Xu, and X. Fu, "D2D fogging: An energy-efficient and incentive-aware task offloading framework via network-assisted D2D collaboration," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3887–3901, Dec. 2016.

[21] N. Fan, X. Wang, D. Wang, Y. Lan, and J. Hou, "A collaborative task offloading scheme in D2D-assisted fog computing networks," in *Proc. IEEE WCNC*, May 2020, pp. 1–6.

[22] U. Saleem, Y. Liu, S. Jangsher, X. Tao, and Y. Li, "Latency minimization for D2D-enabled partial computation offloading in mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 69, no. 4, pp. 4472–4486, Apr. 2020.

[23] Q. Lin, F. Wang, and J. Xu, "Optimal task offloading scheduling for energy efficient D2D cooperative computing," *IEEE Commun. Lett.*, vol. 23, no. 10, pp. 1816–1820, Oct. 2019.

[24] H. Ko, J. Lee, and S. Pack, "Spatial and temporal computation offloading decision algorithm in edge cloud-enabled heterogeneous networks," *IEEE Access*, vol. 6, pp. 18920–18932, 2018.

[25] B. Zhou, R. N. Calheiros, S. N. Srirama, and R. Buyya, "mCloud: A context-aware offloading framework for heterogeneous mobile cloud," *IEEE Trans. Services Comput.*, vol. 10, no. 5, pp. 797–810, Sep./Oct. 2017.

[26] T. Zhao, S. Zhou, L. Song, Z. Jiang, X. Guo, and Z. Niu, "Energy-optimal and delay-bounded computation offloading in mobile edge computing with heterogeneous clouds," *China Commun.*, vol. 17, no. 5, pp. 191–210, May 2020.

[27] Z. Wu, L. Gui, J. Chen, H. Zhou, and F. Hou, "Mobile cloudlet assisted computation offloading in heterogeneous mobile cloud," in *Proc. WCSP*, Oct. 2016, pp. 1–6.

[28] H. Jin, X. Zhu, and C. Zhao, "Computation offloading optimization based on probabilistic SFC for mobile online gaming in heterogeneous network," *IEEE Access*, vol. 7, pp. 52168–52180, 2019.

[29] Y. Wu, B. Shi, L. P. Qian, F. Hou, J. Cai, and X. S. Shen, "Energy-efficient multi-task multi-access computation offloading via NOMA transmission for IoTs," *IEEE Trans. Ind. Informat.*, vol. 16, no. 7, pp. 4811–4822, Jul. 2020.

[30] M. Zhao *et al.*, "Energy-aware task offloading and resource allocation for time-sensitive services in mobile edge computing systems," *IEEE Trans. Veh. Technol.*, vol. 70, no. 10, pp. 10925–10940, Oct. 2021.

[31] "Distributed Training with TensorFlow." TensorFlow. [Online]. Available: https://www.tensorflow.org/guide/distributed_training (Accessed: Oct. 7, 2021).

[32] "AWS Lambda." AWS. [Online]. Available: https://aws.amazon.com/lambda/ (Accessed: Oct. 7, 2021).

[33] H. Ko, S. Pack, and V. C. M. Leung, "Coverage-guaranteed and energy-efficient participant selection strategy in mobile crowdsensing," *IEEE Internet Things J.*, vol. 6, no. 2 pp. 3202–3211, Apr. 2019.

[34] H. Ko, S. Pack, and V. C. M. Leung, "Performance optimization of serverless computing for latency-guaranteed and energy-efficient offloading in energy harvesting Internet of Things," Korea Univ., Seoul, Republic of Korea, Rep. 1, Mar. 2020. [Online]. Available: http://asq.kr/ZTWcN3WPkUext

Authorized licensed use limited to: University of Leeds. Downloaded on May 06,2024 at 17:12:00 UTC from IEEE Xplore. Restrictions apply.

[35] J. Zheng, Y. Cai, X. Shen, Z. Zheng, and W. Yang, "Green energy optimization in energy harvesting wireless sensor networks," *IEEE Commun. Mag.*, vol. 53, no. 11, pp. 150–157, Nov. 2015.

[36] A. Yousefpour, G. Ishigaki, and J. P. Jue, "Fog computing: Towards minimizing delay in the Internet of Things," in *Proc. EDGE*, Honolulu, HI, USA, Jun. 2017, pp. 17–24.

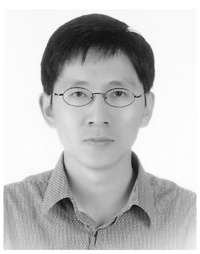
[37] M. Li, "Performance study of wireless powered sensor networks," in *Proc. IEEE IoTaIS*, Bali, Indonesia, Nov. 2019, pp. 1–4.

[38] S.-W. Ko, K. Han, and K. Huang, "Wireless networks for mobile edge computing: Spatial modeling and latency analysis," *IEEE Trans. Wireless Commun.*, vol. 17, no. 8, pp. 5225–5240, Aug. 2018.

[39] "Linear Programming." Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/Linear_programming (Accessed: Oct. 8, 2021).

Haneul Ko (Member, IEEE) received the B.S. and Ph.D. degrees from the School of Electrical Engineering, Korea University, Seoul, South Korea, in 2011 and 2016, respectively.

He is currently an Assistant Professor with the Department of Computer and Information Science, Korea University. From 2017 to 2018, he was with the Smart Quantum Communication Research Center, Korea University and a Visiting Postdoctoral Fellow with the University of British Columbia, Vancouver, BC, Canada. From 2016 to 2017, he was a Postdoctoral Fellow of Mobile Network and Communications, Korea University. His research interests include 5G networks, network automation, mobile cloud computing, SDN/NFV, and future Internet.



Sangheon Pack (Senior Member, IEEE) received the B.S. and Ph.D. degrees in computer engineering from Seoul National University, Seoul, South Korea, in 2000 and 2005, respectively.

In 2007, he joined the Faculty of Korea University, Seoul, where he is currently a Professor with the School of Electrical Engineering. From 2005 to 2006, he was a Postdoctoral Fellow with the Broadband Communications Research Group, University of Waterloo, Waterloo, ON, Canada. His research interests include softwarized networking (SDN/NFV), 5G/6G mobile core networks, mobile-edge computing/programmable data plane, and vehicular networking.

Prof. Pack was the recipient of the IEEE/Institute of Electronics and Information Engineers Joint Award for IT Young Engineers Award 2017, the Korean Institute of Information Scientists and Engineers Young Information Scientist Award 2017, Korea University TechnoComplex Crimson Professor 2015, the Korean Institute of Communications and Information Sciences Haedong Young Scholar Award 2013, LG Yonam Foundation Overseas Research Professor Program in 2012, and the IEEE ComSoc APB Outstanding Young Researcher Award in 2009. He served as a TPC Vice-Chair for information systems of IEEE WCNC 2020, a Track Chair of IEEE VTC 2020-Fall/2010-Fall and IEEE CCNC 2019, a TPC Chair of IEEE/IEIE ICCE-Asia 2018/2020, EAI Qshine 2016, and ICOIN 2020, a Publication Co-Chair of IEEE INFOCOM 2014 and ACM MobiHoc 2015, a Symposium Chair of IEEE WCSP 2013, a TPC Vice-Chair of ICOIN 2013, and a Publicity Co-Chair of IEEE SECON 2012. He is an Editor of IEEE INTERNET OF THINGS JOURNAL, *Journal of Communications Networks*, *IET Communications*, and he was a Guest Editor of IEEE TRANSACTIONS ON EMERGING TOPICS IN COMPUTING and IEEE TRANSACTIONS ON NETWORK SCIENCE AND ENGINEERING.

Victor C. M. Leung (Life Fellow, IEEE) received the B.A.Sc. (Hons.) and Ph.D. degrees in electrical engineering from the University of British Columbia (UBC), Vancouver, BC, Canada, in 1977 and 1982, respectively.

He attended graduate school at UBC on a Canadian Natural Sciences and Engineering Research Council Postgraduate Scholarship. From 1981 to 1987, he was a Senior Member of Technical Staff and Satellite System Specialist with MPR Teltech Ltd., Burnaby, BC, Canada. In 1988, he was a Lecturer with the Department of Electronics, Chinese University of Hong Kong, Hong Kong. He returned to UBC as a Faculty Member in 1989 and held the positions of a Professor and a TELUS Mobility Research Chair of Advanced Telecommunications Engineering with the Department of Electrical and Computer Engineering, when he retired at the end of 2018 and became a Professor Emeritus. Since March 2019, he has been appointed as a Distinguished Professor with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China. He has coauthored more than 1300 journal articles and conference papers, 46 book chapters, and co-edited 14 book titles. Several of his papers had been selected for best paper awards. According to Google Scholar, his works have been cited more than 37 000 times for an H-index of 88. His research interests are in the broad areas of wireless networks and mobile systems.

Dr. Leung was awarded the APEBC Gold Medal as the Head of the graduating class in the Faculty of Applied Science. He received the IEEE Vancouver Section Centennial Award, the 2011 UBC Killam Research Prize, the 2017 Canadian Award for Telecommunications Research, the 2018 IEEE TCGCC Distinguished Technical Achievement Recognition Award, and the 2018 ACM MSWiM Reginald Fessenden Award. He co-authored papers that won the 2017 IEEE ComSoc Fred W. Ellersick Prize, the 2017 IEEE Systems Journal Best Paper Award, the 2018 IEEE CSIM Best Journal Paper Award, and the 2019 IEEE TCGCC Best Journal Paper Award. He is a registered Professional Engineer in the Province of British Columbia, Canada. He is serving on the Editorial Boards for the IEEE TRANSACTIONS ON GREEN COMMUNICATIONS AND NETWORKING, IEEE TRANSACTIONS ON CLOUD COMPUTING, IEEE ACCESS, *IEEE Network*, *Computer Communications*, and several other journals, and has previously served on the editorial boards of the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS–Wireless Communications Series and Series on Green Communications and Networking, IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE TRANSACTIONS ON COMPUTERS, IEEE WIRELESS COMMUNICATIONS LETTERS, and *Journal of Communications and Networks*. He has guest edited many journal special issues, and provided leadership to the organizing committees and technical program committees of numerous conferences and workshops. He is named in the current Clarivate Analytics list of Highly Cited Researchers. He is a Fellow of the Academy of Science of the Royal Society of Canada, the Engineering Institute of Canada, and the Canadian Academy of Engineering. He was a Distinguished Lecturer of the IEEE Communications Society.

Authorized licensed use limited to: University of Leeds. Downloaded on May 06,2024 at 17:12:00 UTC from IEEE Xplore. Restrictions apply.