applied sciences

MDPI

Article

Online Service-Time Allocation Strategy for Balancing Energy Consumption and Queuing Delay of a MEC Server

Jaesung Park [1] and Yujin Lim [2,*]

[1] School of Information Convergence, Kwangwoon University, Seoul 01897, Korea; jaesungpark@kw.ac.kr
[2] Department of IT Engineering, Sookmyung Women's University, Seoul 04310, Korea
* Correspondence: yujin91@sookmyung.ac.kr; Tel.: +82-2-2077-7305

Abstract: MEC servers (MESs) support multiple queues to accommodate the delay requirements of tasks offloaded from end devices or transferred from other MESs. The service time assigned to each queue trades off the queue backlog and energy consumption. Because multiple queues share the computational resources of a MES, optimally scheduling the service time among them is important, reducing the energy consumption of a MES and ensuring the delay requirement of each queue. To achieve a balance between these metrics, we propose an online service-time allocation method that minimizes the average energy consumption and satisfies the average queue backlog constraint. We employ the Lyapunov optimization framework to transform the time-averaged optimization problem into a per-time-slot optimization problem and devise an online service-time allocation method whose time complexity is linear to the number of queues. This method determines the service time for each queue at the beginning of each time slot using the observed queue length and expected workload. We adopt a long short-term memory (LSTM) deep learning model to predict the workload that will be imposed on each queue during a time slot. Using simulation studies, we verify that the proposed method strikes a better balance between energy consumption and queuing delay than conventional methods.

Keywords: mobile edge computing; online resource allocation; task arrival prediction; long short-term memory; Lyapunov optimization

Citation: Park, J.; Lim, Y. Online Service-Time Allocation Strategy for Balancing Energy Consumption and Queuing Delay of a MEC Server. *Appl. Sci.* **2022**, *12*, 4539. https://doi.org/10.3390/app12094539

Academic Editor: Agostino Forestiero

Received: 11 March 2022
Accepted: 28 April 2022
Published: 29 April 2022

1. Introduction

A key characteristic of 5G systems is that most data are generated and consumed locally [1]. To accommodate this characteristic, a multi-access edge computing (MEC) system should be an integral part of 5G systems [2–4]. By enabling devices to offload their tasks to nearby MEC servers (MESs) deployed at the edge of a network, a MEC system is expected to reduce the task service delay. In addition, a MEC system can decrease the traffic load to be imposed on the backhaul network if devices use a cloud server in a data center located far from the devices.

The offloading decision problem that determines whether a device offloads a task to a MES or processes it locally has drawn much attention from researchers. To save the energy of a device and reduce the task completion time, various methods for a device to make an offloading decision have been proposed [5–9]. After a device offloads a task to a MES, it can move out of the service area of the MES. Then, the delay requirement of the task is likely to be violated. To resolve this issue, task migration methods have been proposed in the literature [10,11]. They attempt to overcome the limited coverage of a MES and support the user mobility without task QoS degradation.

Once a MES receives tasks from devices or other MESs, the MES processes the tasks by appropriately managing its computation and storage resources. The methods managing the resources of a MES mainly focus on solving the MES congestion problem to increase the utility of a MEC system. In other words, they attempt to prevent a MES from being congested by balancing loads among MESs or transferring tasks from a congested MES

to a cloud server. The load-balancing methods proposed in [12,13] redistribute loads among MESs by driving devices to offload their tasks to the MES that can serve them the best. However, the load of a MES is influenced not only by the tasks offloaded directly from devices, but also the tasks migrated from other MESs. Thus, in [14], the authors attempted to balance the loads among MESs by moving the tasks in an overloaded MES to an underloaded MES. A MES congestion problem can also be resolved by an efficient collaboration between a MES and a cloud server [15,16]. A MES determines whether to process a task or offload it to a cloud server to reduce overall energy consumption and latency while satisfying the task's service requirement [15]. The authors in [16] proposed a task offloading method at a MES to minimize the task processing cost of a MES while guaranteeing queue stability.

Previous works on MEC system resource management considered all the resources in a path from a device to a MES. Thus, the formulated problem involves not only the local information, but also the information of the other nodes. Sometimes, they even require the system-wide global information. Then, it is assumed implicitly or explicitly that all the information needed to resolve the formulated problem is known to a device or a MES whenever it makes a resource management decision. However, in a practical MEC system, the nodes in a MEC system are supposed to be autonomous. Thus, the state information of one node is hardly known to the other node. For example, in cellular networks, user devices do not exchange information among themselves. Moreover, a user device and a MES are recommended not to exchange their state information for security reasons. Therefore, to implement a practical MEC system, it is important for a node to make a resource management decision by using only its local information.

As noted in [17], there are various MEC application scenarios and each application task has its specific QoS requirements. It has been assumed in the literature that a MES provides a separate FIFO queue for each device. A device may offload several tasks having different QoS requirements. Then, to satisfy the specific QoS requirement of each task, a MES has to maintain a separate queue for each task offloaded by each device and exert a sophisticated computation scheduling method. Thus, the resource management burden of a MES increases as the number of devices and the number of tasks with different QoS requirements increase. To resolve such an issue, the class-based QoS concept has often been adopted in wireless networks and the Internet. The basic idea of the class-based QoS concept is to group various applications into a set of classes according to their QoS characteristics and guarantee the QoS specified for each class, instead of handling the specific QoS requirement of each application task separately. For example, the LTE network assigns a QoS class identifier to each carrier type. Various application services are classified according to their priority, packet error rate, and packet delay budget [18]. Applications with similar QoS requirements in terms of the priority, packet error rate, and packet delay budget are classified into the same class. A similar concept is adopted in the Internet as the DiffServ service classes [19]. In this paper, we follow the class-based QoS concept. A MES classifies tasks according to their delay requirements and manages a queue for each class. A MES manages the service-time allocated to each queue to provide differentiated services among the queues in terms of the average queuing latency.

Another important criterion that must be considered in devising a MES resource management method is the energy consumption of a MES. The amount of energy consumed by a MES to serve a set of class queues must be minimized to reduce both the operational expenditure and the amount of carbon dioxide emissions. A MEC system is expected to reduce the energy consumption of a device. Therefore, the issue of minimizing the energy consumption of a device has been inspected thoroughly. However, comparatively, the problem of minimizing the energy consumption of a MES has not been explored extensively. Thus, in this paper, we address the service-time allocation problem of a MES in the situation where the task input process to a MES varies in a probabilistic way and its distribution is not known in advance. In this environment, we devise a service-time allocation strategy by using only the task queue length and the recent sequences of

the task input rates, which can be measured by a MES without any aid of other nodes. Specifically, we aim to minimize the time-averaged energy consumption of a MES, which is not overloaded while guaranteeing the average task queue length below a given threshold for all the queues. The contributions of this study are as follows:

- We consider a service-time allocation problem at a MES. We formulate a time-averaged energy-minimization problem with an average queuing delay constraint. The formulated problem is complex because it involves information on the input dynamics for future time slots when determining an optimal allocation strategy. Thus, using the Lyapunov approach, we transform it into a per-time-slot optimization problem and devise an online service time allocation method.
- Unlike other Lyapunov-based methods that use only the observed queue length at the beginning of a time slot, we explicitly consider the dynamics of an input process by predicting the amount of workload that will be imposed on each queue during each time slot. For the prediction, we use a long short-term memory (LSTM) model [20], which is a deep learning model widely used in predicting time series data [21,22]. Through simulation studies, we show that we can strike a better balance between the average energy consumption and average queuing delay than with a conventional Lyapunov-based method.
- We devise a service-time allocation algorithm whose complexity is $O(|N|)$, where N is the set of queues managed by a MES. Most optimization-based methods use an iterative process to resolve the formulated problem. However, the iterative process requires time to converge. For example, the convergence time of a subgradient method with a constant step size is known to be $O(1/\epsilon)$ [23], where $0 < \epsilon \ll 1$ is the difference between the termination point and an optimal point. Thus, the time complexity of the conventional method is $O(|N|/\epsilon)$, which is greater than that of ours.
- By providing a weight parameter on the importance of the energy consumption, our method enables a MEC system operator to control the correct balance between the energy consumption and queuing delay according to his/her operational purpose.

The remainder of this paper is organized as follows. In Section 2, we present the related works. In Section 3, we describe the system model and formally state the service-time allocation problem. In Section 4, we present our online service-time allocation method. After validating the proposed method by comparing its performance to those of conventional methods in Section 5, we conclude the paper with future research directions in Section 6. We present the notations used in this paper in Abbreviations for the readers' convenience.

2. Related Works

2.1. Resource Management Methods in a MEC System

The resource management problem in a MEC system has been investigated in various scenarios. In [6], the authors formulated a long-term system energy-minimization problem with average end-to-end delay constraints in a MEC system composed of one MES, multiple APs, and multiple devices, all exploiting low-power sleep mode. By using the Lyapunov optimization framework, they separated the problem into a CPU scheduling problem at a MES and a device–AP association problem at a device. They solved the first problem by an iterative algorithm and the second problem based on multi-agent reinforcement learning (MARL). In [7], an optimal secondary sensing strategy for saving the energy consumption of a device was designed in an environment where devices opportunistically use a licensed spectrum as the secondary users. A cost function was defined as the sum of the total energy consumption of the devices and the total queue length, which is the sum of the queue length of the devices and the queue length at a MES for each device. Each device employs a deep Q network (DQN) agent to minimize the long-term average cost by making a sensing decision for each channel and selecting both the local CPU frequency and the transmission power. In [8], the authors studied the task offloading problem and the resource allocation problem in multiple devices and a multi-edge network scenario. To reduce the resource cost for task offloading and improve the utilization of server resources, they proposed an

online predictive offloading algorithm based on deep reinforcement learning (DRL) and LSTM. LSTM was used to predict the next arriving task, which was used by a DQN agent in a device for making an offloading decision. When a device decides to offload a task, the DQN agent also determines an optimal transmission power to minimize the offloading cost. However, in these works [6–8], a deep learning agent resides in a device and jointly determines its local decision parameters and the parameters related to the service-time allocation in a MES. Therefore, a device requires the local information, the information in a MES, and the global information to determine the optimal parameters. However, it is not clear how an agent in a device acquires the information of the other nodes and how the decision made by a device is transferred to a MES. In addition, they are device-centric methods. Since decisions are made by a device, their main goal is not to minimize the energy consumption of a MES, but to minimize the energy consumption of a device.

MES-centric methods were proposed in [9,16]. The authors in [16] considered a MEC system composed of one cloud server, one MES, and multiple mobile devices. They focused on the task offloading problem from a MES to a cloud server to minimize the time-averaged penalty. They cast the resource allocation problem at a MES as a Lyapunov optimization problem and proposed a DRL-based approach to solve the problem by incorporating the queue stability constraint into the DRL formulation. They devised a DRL agent by using a soft actor–critic (SAC) algorithm, which learns an optimal policy for resource allocation at a MES while stabilizing the queues. By using the agent, a MES optimally determines whether to offload a task to a cloud server or process the task by itself under the task queue stability condition. Basically, the problem addressed by the authors is a MES congestion resolution problem. In this paper, we deal with the energy consumption minimization problem when a MES is not overloaded. However, we note that our method can work with their method. When a MES detects an impending congestion while using our method, it can invoke the DRL agent to offload some tasks to a cloud server. In [9], an optimal scheduling problem was formulated, which aimed to minimize the long-term average weighted sum of energy consumption and delay of all mobile devices in an environment with one 5G small-cell BS with multiple antennas, one MES, and multiple mobile devices. Since the formulated problem requires information that is generally not available in a practical system, the authors proposed a DDPG-based solution that learns an optimal task offloading decision from mobile devices to a MES, the transmission power of each mobile device, and CPU frequencies allocated by a MES to each task. However, to lean an optimal policy, an agent still needs system-wide information such as the tasks selected by all mobile devices and the channel condition between a BS and all mobile devices. In addition, the agent has to distribute the result of the offloading and resource allocation decision to all the mobile devices so that they take actions accordingly. Since the signaling overhead increases with the number of mobile devices, it may cause a scalability problem. Furthermore, the signaling procedures between devices and a MES is not clear.

2.2. Technical Approaches

A resource management problem in a MEC system is often formulated as an optimization problem with constraints. Although the specific system models are different, the formulated optimization problems are often NP-hard or NP-complete. Various technical approaches were adopted to reduce the complexity of the formulated problem. The authors in [24] cast the virtual network function (VNF) instance deployment and user request assignment problem as an integer linear programming problem and proposed two heuristic algorithms with limited complexity. In [14], the authors proposed a task-redirection method to balance the loads among MESs. By lexicographically minimizing the MES load vector, they increased the resource efficiency of a MEC system and reduced the average task blocking rate. In [25], a utility maximization problem was formulated to jointly optimize the video segment cache, transcoding strategy, and wireless resource-allocation strategy. Because this is a mixed-integer nonlinear programming problem, they decomposed the original problem into multiple simpler subproblems and proposed a low-complexity heuristic

method to resolve each subproblem. To simplify the objective function, auxiliary variables were used in [26] to minimize the maximum task execution delay. Bilevel programming and matching algorithms were used in [27] to optimize the task assignment, power allocation, and user association jointly. To reduce the complexity of the formulated problem, these methods transform the original problem into a more tractable form by simplifying the objective function or decomposing it into multiple simpler subproblems. They then devised an iterative process to find an optimal solution to the converted problem. However, the iterative process requires time before finding an optimal solution. Thus, applying them to make online resource-allocation decisions may be inappropriate.

Game theory is also used for resource management in MEC systems [28–30]. Methods based on game theory are scalable and easily adapt to unexpected behaviors. However, implementing a lightweight algorithm with limited complexity using theoretical game models is not easy [31]. For example, a cooperative game requires information exchanges among players, which incurs high signaling costs in a MEC system. Machine learning models have been used to resolve resource management problems. In [32], three regression models were used to obtain an optimal subcarrier allocation. However, because they used an exhaustive search to find optimal labels when constructing training datasets, the computational complexity of their method was high. In addition, they did not consider the input load dynamics or queuing delay when making a resource allocation decision. Therefore, the performance achieved using these methods is not expected to exceed a certain threshold. In [33], a deep deterministic policy gradient (DDPG) managed the available spectrum, computing, and caching resources for MEC-mounted base stations and unmanned aerial vehicles. DDPG was also used in [34] to solve the optimal user association and video quality selection problems. However, DDPG is sensitive to hyperparameters and often converges to a poor solution [35]. The authors in [36] used the Lyapunov optimization approach to optimally offload tasks in a dense cloud radio access network (RAN). The methods using the Lyapunov approach manage resources using only the observed queue length at the start of each time slot. However, because they do not consider input dynamics, these methods are suboptimal, in particular when the input load changes widely over time.

3. System Model and Problem Formulation

Figure 1 shows the system model that we consider in this paper. Tasks arriving at a MES can be classified into two groups. One group is composed of the tasks offloaded directly from devices, and the other group comprises the tasks sent by other MESs by means of the task migration and the load-balancing. In this paper, we focus on the service-time allocation problem occurring when a MES serves the received tasks. Therefore, we do not distinguish the former from the latter. We assumed that once a MES receives a task, it classifies the task according to its type and buffers its workload in a queue arranged for that type of task. To devise an efficient service-time allocation strategy in a MES, we considered a MES that supports a set N of task classes. The time is divided into equally sized slots. We denote τ as the length of a time slot and f_{mes} as the CPU frequency of a MES. A task imposes a workload in terms of the number of CPU cycles required to complete the task. For each task class i, a MES maintains a queue i, which stores the workloads imposed by the class tasks i. We assumed a stochastic fluid queue for each class [37]. That is, we considered the workload imposed by a task to be a continuous fluid, although the workload is a discrete unit. Thus, we do not delineate task boundaries in each queue. We denote the workload stored in queue i at the start of time slot t as $Q_i(t)$.

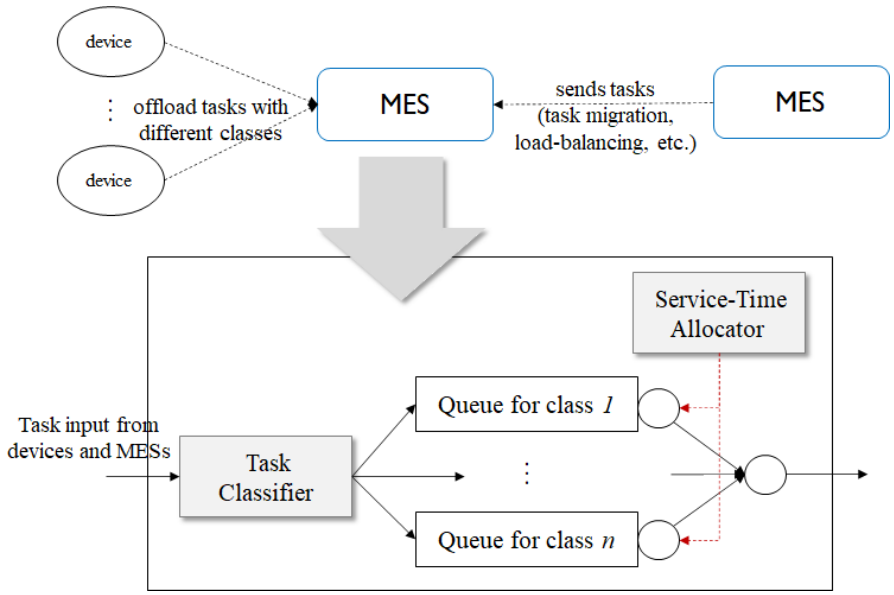


Figure 1. System model.

During each time slot t, the CPU of a MES is shared by the set of queues. If we denote the proportion of time that a MES serves a queue i during a time slot t as $\alpha_i(t)$, the number of CPU cycles used to serve a queue i becomes

$$b_i(t) = \alpha_i(t)\tau f_{mes}, \quad 0 \leq \alpha_i(t) \leq 1, \forall i \in N, \forall t > 0. \tag{1}$$

We adopted a popular cubic model to quantify the energy consumption of a MES [38]. Thus, during time slot t, the amount of energy consumed by a MES to serve queue i is

$$e_i(t) = \alpha_i(t)\tau\kappa f_{mes}^3, \tag{2}$$

where κ denotes the effective switching capacity of the processor. Therefore, the amount of energy consumed by a MES during time slot t is obtained as:

$$E(t) = \sum_{i \in N} e_i(t). \tag{3}$$

If we denote the workload newly imposed on queue i during time slot t as $w_i(t)$, the dynamics of $Q_i(t)$ can be described as

$$Q_i(t+1) = \max\{0, Q_i(t) - b_i(t) + w_i(t)\}. \tag{4}$$

According to Little's law, if the data arrival rate to a stationary queuing system is given as $\bar{w}_i = \mathbb{E}[w_i(t)/\tau]$, the long-term time-averaged queuing latency is given as

$$\bar{L}_i = \lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}[\frac{Q_i(t)}{\bar{w}_i}]. \tag{5}$$

$\bar{L}_i$ is proportional to the long-term time-averaged queue length. To provide differentiated services among the queues in terms of the queuing latency, the long-term time-averaged queue length should be maintained under the queue length threshold $\delta_i = \bar{w}_i \bar{L}_i$. That is,

$$\lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}[Q_i(t)] \leq \delta_i, \quad \forall i \in N. \tag{6}$$

Since δ_i is determined by the delay characteristics of the tasks belonging to the class i, we assumed that δ_i is pre-defined for each class queue.

As seen in Equations (2) and (4), $e_i(t)$ increases with $\alpha_i(t)$, while $Q_i(t)$ decreases as $\alpha_i(t)$ increases. Thus, to strike the correct balance between energy consumption and queuing delay, we require a systematic method to determine $\alpha_i(t)$. To address the trade-off between the energy consumption and queuing delay, we intend to minimize the long-term time-averaged energy consumption of a MES while satisfying the queue length constraint in Equation (6) by optimally controlling the proportion of time that a MES serves each queue. We formulated our problem as the following optimization problem.

$$\text{P1}: \min \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}[E(t)] \tag{7}$$

$$s.t. \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}[Q_i(t)] \leq \delta_i, \quad \forall i \in N. \tag{8}$$

$$0 \leq \alpha_i(t) \leq 1, \quad \forall i \in N. \tag{9}$$

$$\sum_{i\in N} \alpha_i(t) \leq 1. \tag{10}$$

Problem P1 is difficult to solve because we cannot know the statistics of the task arrivals for all time slots when we must determine $\alpha_i(t)$s. To tackle this difficulty, we adopted the Lyapunov optimization framework and a deep learning model to devise an online service-time allocation method that determines $\Omega(t) = \{\alpha_i(t) : i \in N\}$ at the beginning of each time slot t.

4. Online Resource Allocation Method

In this section, we convert Problem P1 into a per-time-slot optimization problem using the Lyapunov optimization framework. We then design an online service-time allocation strategy that can solve the converted problem in $O(|N|)$.

4.1. Per-Time-Slot Optimization Problem

To control the delay constraint in P1, we define the virtual queue $Z_i(t)$ for each task queue that evolves as follows:

$$Z_i(t+1) = \max[0, Z_i(t) + Q_i(t+1) - \delta_i], \quad \forall i \in N. \tag{11}$$

A virtual queue is said to be mean rate stable if

$$\lim_{T\to\infty} \frac{\mathbb{E}[Z_i(T)]}{T} = 0. \tag{12}$$

From the Lyapunov optimization framework in [39], the delay constraint in P1 is satisfied when the virtual queue is mean rate stable. To guarantee the mean rate stability of the virtual queue, we define the following Lyapunov function:

$$L(Z(t)) = \frac{1}{2} \sum_{i\in N} Z_i(t)^2, \quad \forall t. \tag{13}$$

We also define the Lyapunov drift, which is the conditional expected changes in the Lyapunov function over one time slot.

$$\Delta L(Z(t)) = \mathbb{E}[L(Z(t+1)) - L(Z(t))|Z(t)]. \tag{14}$$

By minimizing $\Delta L(Z(t))$ for each time slot, we can make $Z_i(t)$s mean rate stable. However, minimizing only $\Delta L(Z(t))$ may result in unnecessary energy consumption. Thus,

following the approach in [39], we define the drift-plus-penalty function $\Delta_p L(Z(t))$ to consider the energy consumption of a MES.

$$\Delta_p L(Z(t)) = \Delta L(Z(t)) + V\mathbb{E}[E(t)|Z(t)], \tag{15}$$

where V is a weight parameter that controls the importance of the MES energy consumption on $\Delta_p L(Z(t))$. For example, as V increases, the energy consumption becomes more important than the delay constraint for each queue. From the concept of opportunistically minimizing the expectation, our problem becomes minimizing $\Delta_p L(Z(t))$ for each time slot. To achieve this goal, we introduce Proposition 1, which provides the upper bound of $\Delta_p L(Z(t))$.

Proposition 1. *$\Delta_p L(Z(t))$ is upper bounded by*

$$\Delta_p L(Z(t)) \leq B + \mathbb{E}[\sum_{i\in N} b_i(t)^2 - b_i(t)(2Q_i(t) + 2w_i(t) + Z_i(t)) + Vb_i(t)\kappa f_{mes}^3 | Z(t)], \tag{16}$$

where $b_i(t) = \alpha_i(t)\tau f_{mes}$ and $B = \sum_{i\in N} Q_i(t)^2 + w_i(t)^2 + 2Q_i(t)w_i(t) + Z_i(t)(Q_i(t) + w_i(t)) + \delta_i^2$ is a positive constant irrelevant to $\alpha_i(t)$s.

Proof. Since $Z_i(t+1) = \max[0, Z_i(t) + Q_i(t+1) - \delta_i]$, $Z_i(t) \leq Z_i(t)^2 + 2Z_i(t)(Q_i(t+1) - \delta_i) + (Q_i(t+1) - \delta_i)^2$. Let $\Delta_{Z_i(t)} = \frac{1}{2}(Z_i(t+1)^2 - Z_i(t)^2)$. Then,

$$\Delta_{Z_i(t)} \leq \frac{(Q_i(t+1) - \delta_i)^2}{2} + Z_i(t)(Q_i(t+1) - \delta_i).$$

Since $(x+y)^2 \leq 2x^2 + 2y^2$,

$$\Delta_{Z_i(t)} \leq Q_i(t+1)^2 + Z_i(t)Q_i(t+1) + \delta_i^2. \tag{17}$$

Because $Q_i(t+1) = \max[0, Q_i(t) - b_i(t) + w_i(t)]$, $Q_i(t+1)^2 \leq Q_i(t)^2 + b_i(t)^2 + w_i(t)^2 + 2Q_i(t)(w_i(t) - b_i(t)) - 2w_i(t)b_i(t)$. By rearranging the terms, we obtain

$$Q_i(t+1)^2 \leq B_i(t) + b_i(t)^2 - 2b_i(t)(Q_i(t) + w_i(t)), \tag{18}$$

where $B_i(t) = Q_i(t)^2 + w_i(t)^2 + 2Q_i(t)w_i(t)$ is a positive constant unrelated to $b_i(t)$. By placing Equation (18) into Equation (17), we obtain

$$\Delta_{Z_i(t)} \leq B_i(t) + b_i(t)^2 - 2b_i(t)(Q_i(t) + w_i(t)) + Z_i(t)(Q_i(t) - b_i(t) + w_i(t)) + \delta_i^2. \tag{19}$$

If we let $C_i(t) = B_i(t) + Z_i(t)(Q_i(t) + w_i(t)) + \delta_i^2$, we obtain:

$$\Delta_{Z_i(t)} \leq C_i(t) + b_i(t)^2 - b_i(t)(2Q_i(t) + 2w_i(t) + Z_i(t)). \tag{20}$$

As $\Delta L(Z(t)) = \mathbb{E}[L(Z(t+1)) - L(Z(t))|Z(t)] = \mathbb{E}[\sum_{i\in N} \Delta_{Z_i(t)}|Z(t)]$,

$$\Delta L(Z(t)) \leq \sum_{i\in N} C_i(t) + \mathbb{E}[\sum_{i\in N} b_i(t)^2 - b_i(t)(2Q_i(t) + 2w_i(t) + Z_i(t))|Z(t)]. \tag{21}$$

Recall that $b_i(t) = \alpha_i(t)\tau f_{mes}^3$ and $e_i(t) = \alpha_i(t)\tau\kappa f_{mes}^3 = b_i(t)\kappa f_{mes}^2$. Because $E(t) = \sum_{i\in N} e_i(t) = \sum_{i\in N} b_i(t)\kappa f_{mes}^2$, we obtain:

$$V\mathbb{E}[E(t)|Z(t)] = V\mathbb{E}[\sum_{i\in N} b_i(t)\kappa f_{mes}^2 | Z(t)]. \tag{22}$$

From Equations (21) and (22), the $\Delta_p L(Z(t))$ is upper bounded as

$$\Delta_p L(Z(t)) \leq B + \mathbb{E}[\sum_{i \in N} b_i(t)^2 - b_i(t)(2Q_i(t) + 2w_i(t) + Z_i(t)) + Vb_i(t)\kappa f_{mes}^2 | Z(t)],$$

where $B = \sum_{i \in N} C_i(t)$ is a positive constant unrelated to $b_i(t)$. □

Note that the space defined in Equation (10) is the subset of the space defined by Equation (9). Then, from Proposition 1 and the concept of opportunistically minimizing the expectation, we can translate Problem P1 as the following per-time-slot optimization problem.

$$\textbf{P2}: argmin_{\Omega(t)=\{\alpha_i(t):i \in N\}} \mathbb{E}[\sum_{i \in N} b_i(t)^2 - b_i(t)(2Q_i(t) + 2w_i(t) + Z_i(t)) + Vb_i(t)\kappa f_{mes}^2].$$

$$s.t. \sum_{i \in N} \alpha_i(t) \leq 1. \quad (23)$$

4.2. Online Service Time Allocation Method

Problem P2 is a quadratic programming problem with a constraint that is not a separable inequality. Therefore, we may use a projected subgradient method to obtain the solution. However, ref. [23] showed that the convergence time of a subgradient method with a constant step size is $O(1/\epsilon)$, where $0 < \epsilon \ll 1$ is the difference between the termination point and the optimal point. Thus, the time complexity of the subgradient method is $O(|N|/\epsilon)$, which may be too long for online decisions. Therefore, in this section, we propose a heuristic algorithm that determines $\Omega(t)$ in $O(|N|)$ time at the beginning of each time slot and is optimal while the sum of the determined $\alpha_i(t)$s is not larger than 1. Let us denote $g_i(t) = b_i(t)^2 - (2Q_i(t)^2 + 2w_i(t)^2 + Z_i(t) - V\kappa f_{mes}^2)b_i(t)$ and ignore Constraint (23). Because $\min(\sum_{i \in N} g_i(t)) \geq \sum_{i \in N} \min(g_i(t))$, we can obtain a solution to Problem P2 by finding an $\alpha_i(t)$ that satisfies $\frac{\partial g_i(t)}{\partial \alpha_i(t)} = 0$ for all $i \in N$. As $b_i(t) = \alpha_i(t)\tau f_{mes}$, $\frac{\partial g_i(t)}{\partial \alpha_i(t)} = 2\tau^2 f_{mes}^2 \alpha_i(t) - \tau f_{mes}(2Q_i(t) + 2w_i(t) + Z_i(t) - V\kappa f_{mes}^2)$. Therefore, if we denote an $\alpha_i(t)$ that minimizes $\partial g_i(t)/\partial \alpha_i(t)$ as $x_i(t)$, it becomes

$$x_i(t) = \frac{2Q_i(t) + 2w_i(t) + Z_i(t) - V\kappa f_{mes}^2}{2\tau f_{mes}}. \quad (24)$$

A MES must decide $\alpha_i(t)$s at the start of each time slot. However, $x_i(t)$ involves $w_i(t)$, which a MES knows only at the end of the time slot t. To resolve this, we adopted an LSTM model to predict $w_i(t)$ at the beginning of a time slot t. To train the LSTM model for each queue, a MES measures the workload imposed during each time slot. At the start of each time slot k, a MES maintains h past workload arrival histories $h_i(k) = \{w_i(k-h), \ldots, w_i(k-1)\}$ for each queue i and feeds $h_i(k)$ to the LSTM model to obtain $\hat{w}_i(k)$ as its output. Because a MES can measure $w_i(k)$ at the end of time slot k, it updates the LSTM model using $(w_i(k) - \hat{w}_i(k))^2$ as the loss function. Once a MES finishes training the LSTM model, it uses the model to predict the amount of workload that will be imposed on each queue during a time slot t at the beginning of time slot t. Specifically, at the start of a time slot t, a MES feeds $(w_i(t-h), \ldots, w_i(t-1))$ to the LSTM and obtains $\hat{w}_i(t)$ as the workload to be imposed on the queue i during time slot t.

By replacing $w_i(t)$ with $\hat{w}_i(t)$ in Equation (24), we obtain:

$$\hat{x}_i(t) = \frac{2Q_i(t) + 2\hat{w}_i(t) + Z_i(t) - V\kappa f_{mes}^2}{2\tau f_{mes}}. \quad (25)$$

Let us denote the fraction of time that a MES allocates to each queue during a time slot t as $\alpha_i^*(t)$. Because $0 \leq \alpha_i(t) \leq 1$, when $\sum_{i \in N} \alpha_i^*(t) \leq 1$, $\alpha_i^*(t)$ is determined as follows.

$$\alpha_i^*(t) = \begin{cases} 1, & 1 \leq \hat{x}_i(t) \\ \hat{x}_i(t), & 0 < \hat{x}_i(t) < 1 \\ 0, & \hat{x}_i(t) \leq 0 \end{cases} \tag{26}$$

If $\sum_{i \in N} \alpha_i^*(t) > 1$, then the optimal solution to P2 is outside the space defined by the constraint in Equation (23). That is, because the workload imposed on queue i (i.e., $Q_i(t) + w_i(t)$) is temporarily larger than the capacity of a MES, a MES cannot make $Q_i(t+1) < \delta_i$. Because the MES resource is insufficient to handle the total workload imposed on it, we adjust the fraction of time assigned to each queue in proportion to $\alpha_i^*(t)$s in Equation (26). Specifically,

$$\alpha_i^{**}(t) = \frac{\alpha_i^*(t)}{\sum_{j \in N} \alpha_j^*(t)}, \quad \forall i \in N. \tag{27}$$

From Equations (26) and (27), a MES determines $\alpha_i(t)$s at the start of each time slot t as follows.

$$\alpha_i(t) = \begin{cases} \alpha_i^*(t), & \sum_{i \in N} \alpha_i^*(t) \leq 1 \\ \alpha_i^{**}(t), & otherwise \end{cases} \tag{28}$$

We summarize our online resource allocation method in Algorithm 1.

Algorithm 1 Online service time allocation algorithm

At the start of each time slot t**:**
  Observe $Q_i(t)$s, $\forall i \in N$.
  Predict $\hat{w}_i(t)$ using an LSTM, $\forall i \in N$.
  $y = 0$.
while $i \leq |N|$ **do**
    $\hat{x}_i(t) = \frac{2Q_i(t) + 2\hat{w}_i(t) + Z_i(t) - V\kappa f_{mes}^2}{2\tau f_{mes}}$.
    $\alpha_i^*(t) = \begin{cases} 1, & 1 \leq \hat{x}_i(t) \\ \hat{x}_i(t), & 0 < \hat{x}_i(t) < 1 \\ 0, & \hat{x}_i(t) \leq 0 \end{cases}$
    y += $\alpha_i^*(t)$
    $i++$
if $y \leq 1$ **then**
    $\alpha_i(t) = \alpha_i^*(t)$
else
    while $i \leq |N|$ **do** $\alpha_i(t) = \frac{\alpha_i^*(t)}{\sum_{j \in N} \alpha_j^*(t)}$

4.3. Properties

We state the characteristics of our algorithm as Proposition 2.

Proposition 2. *If the drift function satisfies the drift condition for all time slots and all possible $Z(t)$ and the expected penalty function $\mathbb{E}[E(t)]$ is lower bounded by a finite value ζ_{min}, the long-term time-averaged expected energy a MES consumes when it uses our online service time allocation algorithm satisfies*

$$\lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[E(t)] \leq \zeta_{opt} + \frac{B}{V}, \tag{29}$$

where ζ_{opt} is the optimal time-averaged penalty value and B a positive constant. In addition, the long-term time-averaged expected virtual queue length satisfies

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{i\in N} \mathbb{E}[|Z_i(t)|] \leq \frac{B + V(\zeta_{opt} - \zeta_{min})}{\epsilon}, \tag{30}$$

where ϵ is a non-negative constant.

Proof. We followed the approach in [39]. Suppose there are constants $B > 0$ and $\epsilon \geq 0$ such that the following drift condition holds for all time slots and possible $Z(t)$s:

$$\Delta L(Z(t)) \leq B - \epsilon \sum_{i\in N} |Z_i(t)|. \tag{31}$$

Let us assume that the expected penalty function $\mathbb{E}[E(t)]$ is lower bounded by a finite value ζ_{min}. Then, for all time slots and all possible resource allocation actions, we have

$$\mathbb{E}[E(t)] \geq \zeta_{min}. \tag{32}$$

We denote the optimal time-averaged penalty value as ζ_{opt}. That is, for all time slots and all possible resource allocation actions, we obtain

$$\mathbb{E}[E(t)] \leq \zeta_{opt}. \tag{33}$$

Then, for a time slot t, the following inequality holds.

$$\Delta L(Z(t)) + V\mathbb{E}[E(t)] \leq B + V\zeta_{opt} - \epsilon \sum_{i\in N} |Z_i(t)|. \tag{34}$$

Taking expectations on both sides and applying the law of iterated expectations for $t = 0, \ldots, T-1$, we have

$$\mathbb{E}[L(Z(T))] - \mathbb{E}[L(Z(0)] + V \sum_{t=0}^{T-1} \mathbb{E}[E(t)] \leq T(B + V\zeta_{opt}) - \epsilon \sum_{t=0}^{T-1} \sum_{i\in N} |Z_i(t)|. \tag{35}$$

Dividing both side of Equation (35) by TV, we obtain

$$\frac{\sum_{t=0}^{T-1} \mathbb{E}[E(t)]}{T} \leq \zeta_{opt} + \frac{B}{V} + \frac{\mathbb{E}[L(Z(0))] - \mathbb{E}[L(Z(T)]}{TV}. \tag{36}$$

Then, by taking $T \to \infty$, we obtain

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[E(t)] \leq \zeta_{opt} + \frac{B}{V}.$$

Because $0 \leq V \sum_{t=0}^{T-1} \mathbb{E}[E(t)] \leq \infty$ and $0 \leq L(Z(T)) \leq \infty$, by dividing both sides of Equation (35) by ϵT and rearranging terms, we also obtain the following inequality.

$$\frac{1}{T} \sum_{t=1}^{T-1} |Z_i(t)| \leq \frac{B + V\zeta_{opt}}{\epsilon} + \frac{V \sum_{t=0}^{T-1} \mathbb{E}[E(t)]}{\epsilon T} + \frac{\mathbb{E}[L(Z(0)) - \mathbb{E}[L(Z(T))]}{\epsilon T} \tag{37}$$

$$\leq \frac{B + V\zeta_{opt}}{\epsilon} + \frac{VT\zeta_{min}}{\epsilon T} + \frac{\mathbb{E}[L(Z(0))]}{\epsilon T} \tag{38}$$

$$= \frac{B + V(\zeta_{opt} - \zeta_{min})}{\epsilon} + \frac{\mathbb{E}[L(Z(0))]}{\epsilon T}. \tag{39}$$

By taking $T \to \infty$, we obtain

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T-1} |Z_i(t)| \leq \frac{B + V(\zeta_{opt} - \zeta_{min})}{\epsilon}. \quad (40)$$

□

5. Performance Evaluation

In this section, we evaluate the performance of the proposed method using simulation studies. The default parameter values are as follows, if not specified otherwise. A MES is equipped with a $f_{mes} = 1.0$ GHz CPU core, whose effective switching capacity is $\kappa = 10^{-27}$. The length of a time slot is $\tau = 1$ s and $V = 10^{15}$. A MES supports $|N| = 3$ classes, whose delay requirement is configured as $\delta_i = 10i$ ms for each $i \in N$. To emulate the task arrival process for each class, we used the traffic traces from a cellular network containing the time that a device generates a call. We used the dataset that results from a computation over the call detail records (CDRs) generated by the Telecom Italia cellular network for nine days from the city [40]. Assuming that the call generation pattern of a device is similar to the task offloading pattern of the device, we divided the real traffic traces into time slots and calculated the number of tasks generated during each time slot. The workload imposed by a task is influenced by the applications to which a task belongs [41]. In this study, for each task class i, the workload imposed by a task is randomly selected in $[w_i^{min}, w_i^{max}]$ according to the uniform distribution. For each class $i \in N$, we set $[w_i^{min}, w_i^{max}] = [i \times 10.0, i \times 31.6]$ kilocycles. To focus on the service-time allocation problem at a MES, we considered the situation where a MES is not overloaded such that $\lim_{T\to\infty} 1/T \sum_{t=1}^{T} \sum_{i\in N} w_i(t) < \tau f_{mes}$.

5.1. Task Arrival Prediction Using LSTM

To predict the number of tasks arriving at each queue during a time slot, we used a single-layer LSTM with five hidden units for a given sequence input. We add Table 1, which shows the important parameters and their values used for the LSTM training.

Table 1. The parameters for LSTM training.

Parameters	Values
Learning rate	0.01
Number of epochs	3000
Input size	3 to 5
Hidden size	5
Activation function	ReLU
Optimization function	Adam

To compare the prediction performance according to the length of the input sequence (i.e., h), we measured the prediction error in terms of $\zeta_i(t) = ((w_i(t) - \hat{w}_i(t))/w_i(t))^2$. Then, we compared the mean and the standard deviation of the root-mean-squared (RMS) prediction error in Table 2. If we denote the number of time steps used to calculate the RMS as n, the RMS for a class i queue is calculated as $\sqrt{\frac{1}{n}\sum_{t=1}^{n} \zeta_i(t)}$.

We observe in Table 2 that the best performance is obtained when the input size is four. Figure 2 shows the differences in the actual and predicted values over time for each class when the input size is four. The accuracy of the predicted values is quite high compared to the actual values.

Table 2. Performance comparison of LSTM prediction in terms of RMS.

	Class 1		Class 2		Class 3	
	Avg.	**Std. Dev.**	**Avg.**	**Std. Dev.**	**Avg.**	**Std. Dev.**
$h = 3$	0.1022	0.0875	0.1182	0.1069	0.0738	0.0662
$h = 4$	0.0957	0.0767	0.1152	0.1016	0.0889	0.0795
$h = 5$	0.1057	0.0836	0.1249	0.0948	0.0931	0.0709

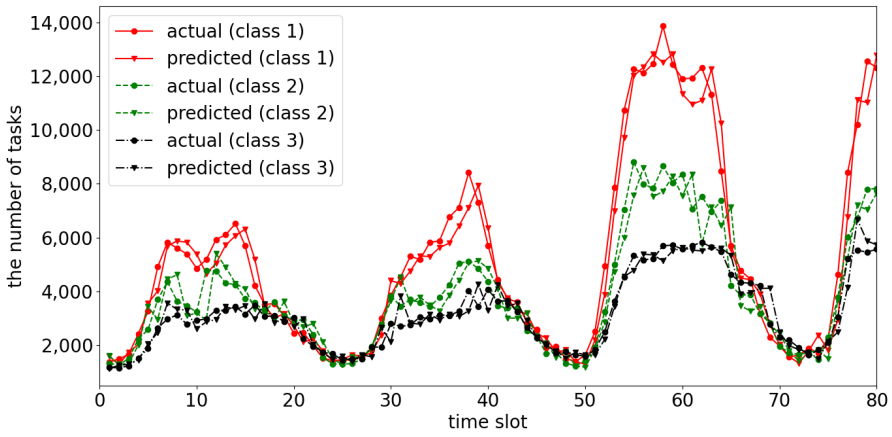


Figure 2. Comparison between actual and predicted values.

5.2. Performance Comparisons

We compared the performance of the proposed method with those of three alternatives. The first alternative is EVEN, which determines $\alpha_i(t) = 1/|N|$. The second alternative determines $\alpha_i(t)$ in proportion to the queue-length threshold δ_i. Specifically, if we denote $\eta_i = 1/\delta_i$, the second alternative determines $\alpha_i(t) = \eta_i / \sum_{j \in N} \eta_j$. Henceforth, we refer to this as TPRA. The third alternative, WoPred, is similar to our method in that it uses a Lyapunov optimization framework. However, WoPred uses only the observed queue length without considering the input workload dynamics when determining $\alpha_i(t)$s at the start of each time slot. We also show the results obtained when our method uses $w_i(t)$s instead of $\hat{w}_i(t)$s, which indicate the upper performance bound of our method.

First of all, we analyzed the time complexity of these methods. Algorithm 1 describes our online resource allocation method. The method is composed of two main parts—the prediction of the workload to be imposed on the queue i during time slot t and the determination of the time fraction that a MES allocates to each queue during a time slot t. To analyze the time complexity of the workload prediction, we considered the time consumed by the LSTM deep learning model to predict the workload. Since the deep learning technique is used for prediction after going through the training phase, the required time for the prediction is $O(1)$. Next, the time complexity of the time fraction determination is $O(N)$, which is proportional to the set of queues managed by a MES. Therefore, the time complexity of our method is $O(N)$. Meanwhile, in the case of the ideal method, the time complexity is $O(N)$ because it has only the time fraction determination process. In *EVEN*, *TPRA*, and *WoPred*, only the time fraction determination process is included. However, the difference among these three methods is in how they determine the fraction of the time that a MES allocates to each queue during a time slot t ($\alpha_i^*(t)$). Since *EVEN* and *TPRA* allocate $\alpha_i^*(t)$s statically, their time complexity is $O(1)$. In contrast, since *WoPred* dynamically allocates service-time at each time slot by calculating $\alpha_i^*(t)$s, its time complexity is $O(N)$.

In Figure 3, we show how each method changes $Q_i(t)$s over time. We can see in Figure 3a that WoPred makes $Q_1(t)$ fluctuate widely and does not make $Q_1(t) < \delta_1$ most of thetime. We also observe similar behaviors in Figure 3b,c. This is because WoPred uses only the current queue length when it allocates service-time for each queue. When

EVEN and TPRA are used, $Q_1(t)$ is maintained under δ_1. However, $Q_2(t)$ and $Q_3(t)$ are not always maintained below δ_2 and δ_3. This is attributed to the fact that they statically allocate service-time. If the workloads imposed on the Class 2 queue and Class 3 queue are temporarily larger than the service-time allocated to them, $Q_2(t)$ and $Q_3(t)$ rise sharply. In this figure, we can observe that our method makes $Q_i(t) < \delta_i$ for all i all the time by dynamically allocating service-time to each queue according to the current queue length and the expected amount of workload.

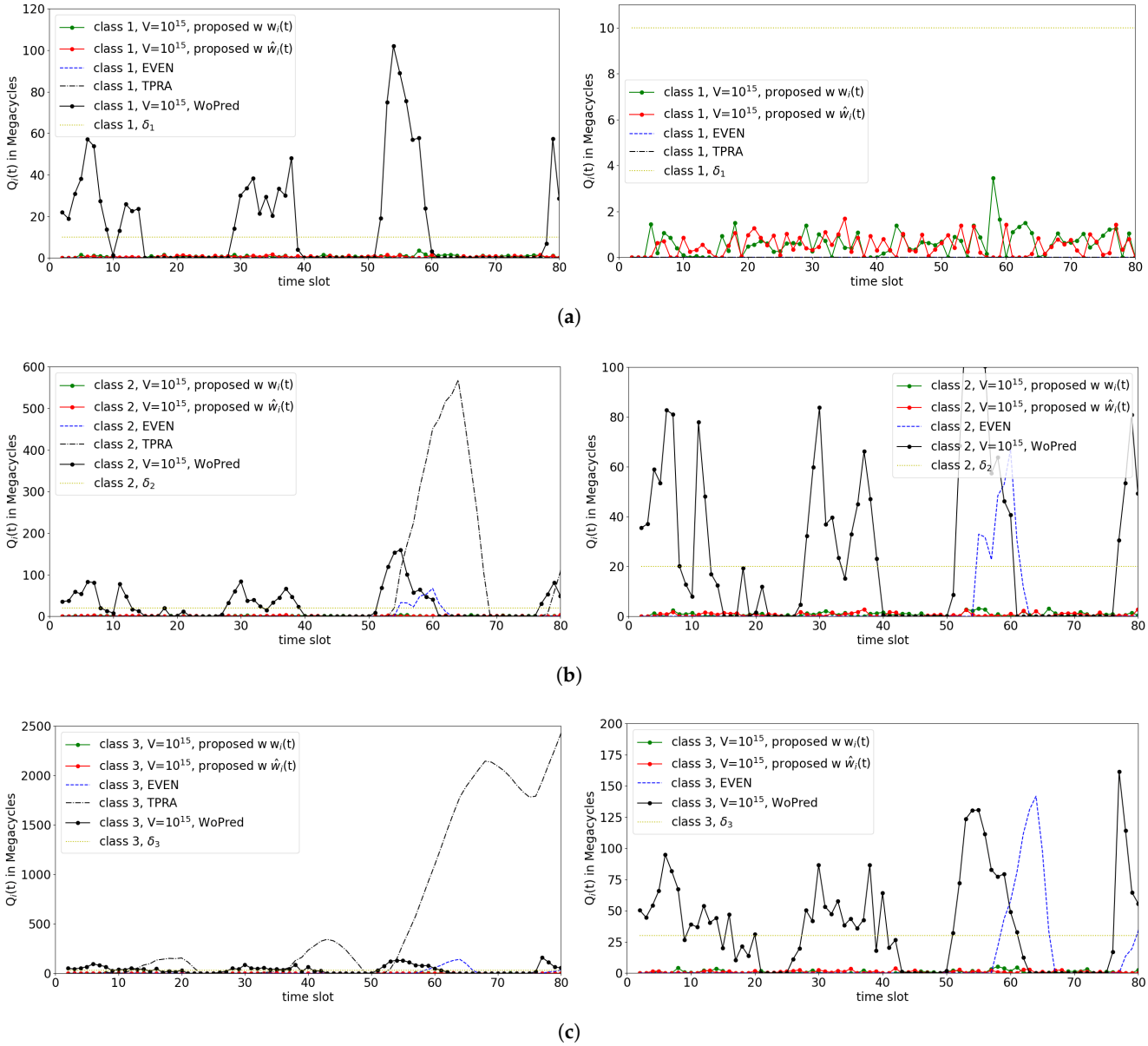
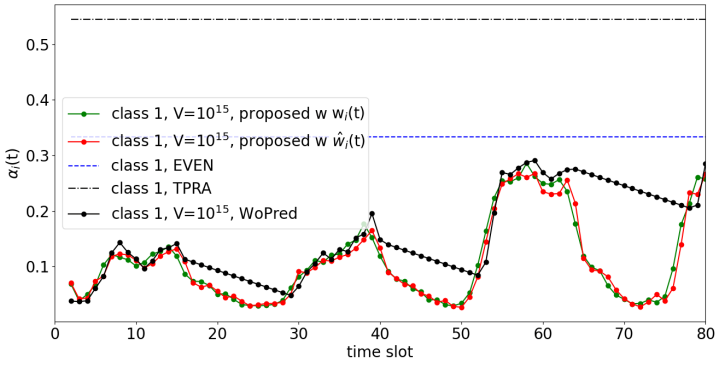
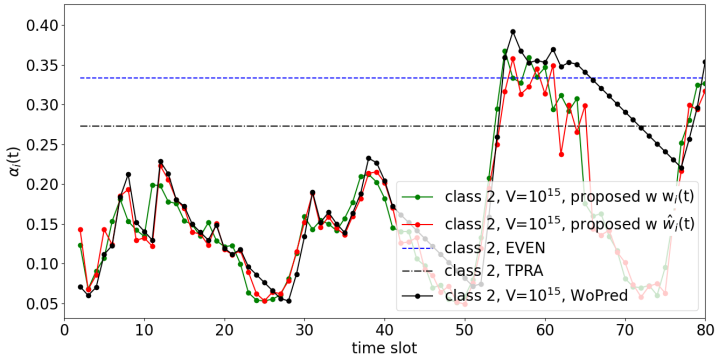


Figure 3. Comparison of $Q_i(t)$ for three classes. (**a**) $Q_1(t)$ over time, (**b**) $Q_2(t)$ over time, and (**c**) $Q_3(t)$ over time.

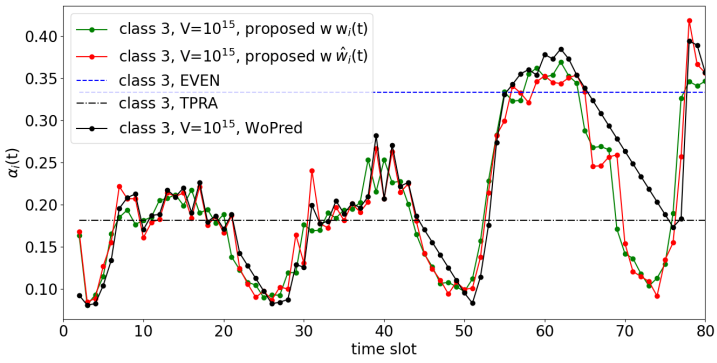
To further examine this phenomenon, we show the changes in $\alpha_i(t)$s over time in Figure 4.



(**a**)



(**b**)



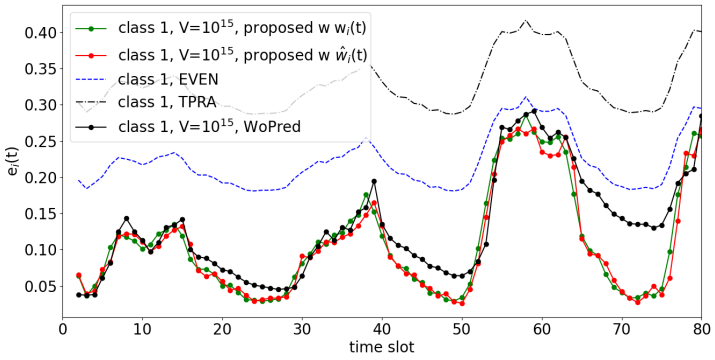
(**c**)

Figure 4. Comparison of $\alpha_i(t)$ for three classes. (**a**) $\alpha_1(t)$ over time, (**b**) $\alpha_2(t)$ over time, and (**c**) $\alpha_3(t)$ over time.

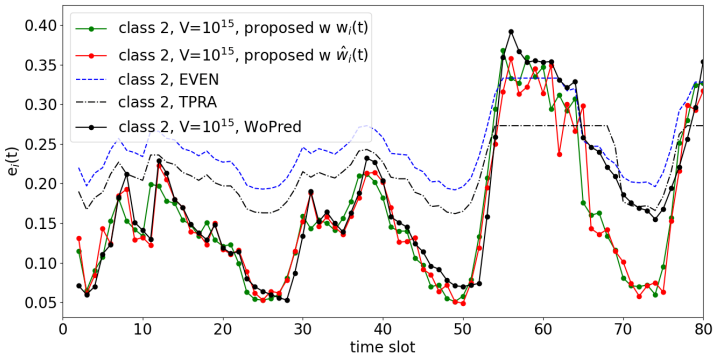
When the workload arriving at each queue is small and the length of each queue is small, all methods maintain $Q_i(t) < \delta_i$ ($\forall i \in N$). However, when the workload fed to a queue i is small and the workload arriving at another queue j is high, the MES must decrease $\alpha_i(t)$ and increase $\alpha_j(t)$ to avoid the temporal overload of queue j. However, EVEN and TPRA statically assign $\alpha_i(t)$s without considering the input workload dynamics and states of the queues. Because they cannot adjust the amount of CPU resources allocated to each queue, they cannot prevent the temporal overload of the queue. By contrast, our method dynamically adjusts $\alpha_i(t)$s according to $Q_i(t)$s, $w_i(t)$s, and δ_i. Because our method can increase $\alpha_j(t)$ more than that allocated by TPRA and decrease $\alpha_i(t)$ less than that allocated by TPRA, it can maintain the queue length below its target threshold.

In Figure 5, we compare $e_i(t)$s over time. We observe that the energy consumed by a MES is high when static methods (EVEN and TPRA) are used. The results are attributed to the manner in which each method determines $\alpha_i(t)$s. We observe in Figure 4 that the static

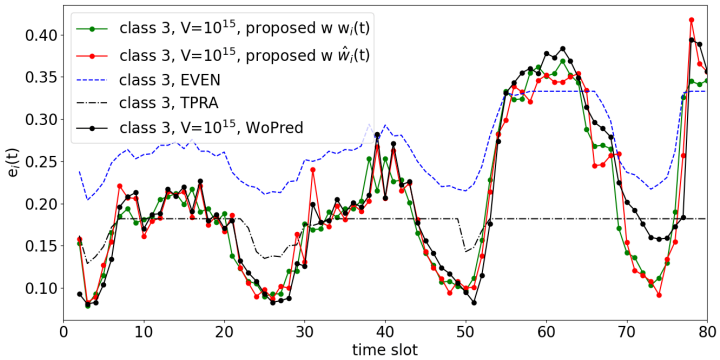
methods assign more service time to each queue than the dynamic methods (our method and WoPred) during times when the queues are not temporarily overloaded. That is, static methods make a MES in an active state unnecessarily longer than dynamic methods. A MES in an active state consumes energy, even when a queue is empty. According to [42], the idle energy consumed by a MES corresponds to 50∼70% of the peak power consumption when a MES is fully used. In contrast, our method and WoPred dynamically determine $\alpha_i(t)$s at the start of each time slot by considering $Q_i(t)$s, reflecting the historical resource allocations. Therefore, a MES can be turned into a low-power mode for $(1 - \sum_{i \in N} \alpha_i(t))\tau$. Thus, they consume less energy than the static methods.



(**a**)



(**b**)



(**c**)

Figure 5. Comparison of $e_i(t)$ for three classes. (**a**) $e_1(t)$ over time, (**b**) $e_2(t)$ over time, and (**c**) $e_3(t)$ over time.

Because WoPred does not consider a workload arrival process (i.e., $\hat{w}_i(t)$), it determines $\alpha_i(t)$s in a reactive manner. Thus, when we examine the $\alpha_i(t)$s obtained by WoPred, the $\alpha_i(t)$s lag because WoPred reacts only to the previous situation reflected in $Q_i(t)$s. That

is, WoPred allocates the $\alpha_i(t)$s using only $Q_i(t)$s without considering $\hat{w}_i(t)$s. Thus, the $\alpha_i(t)$s determined by WoPred may be larger or smaller than the $\alpha_i(t)$s required to maintain $Q_i(t+1) \leq \delta_i$. Note that $Q_i(t+1)$ is affected by $w_i(t)$. Because WoPred considers $w_i(t)$ when it determines $\alpha_i(t+1)$ using $Q_i(t+1)$, WoPred reacts to $w_i(t)$. In contrast, our method predicts $w_i(t)$ at the beginning of a time slot t and considers $\hat{w}_i(t)$ when it determines $\alpha_i(t)$. Thus, by considering $Q_i(t)$ and $\hat{w}_i(t)$ at the start of time slot t, our method can determine $\alpha_i(t)$s more optimally than WoPred, which results in smaller $Q_i(t)$s and $e_i(t)$s than WoPred.

5.3. Parameter Effect

In Figures 6 and 7, we show the average MES energy consumption and an average queue length after the 140th time slot with different Vs. These two figures show the influence of V, which trades off energy consumption and queue length. V determines the importance of the penalty term (i.e., a MES energy consumption) in the drift-plus-penalty function. Thus, as V increases, the amount of energy consumed by a MES dominates $\Delta_p L(Z(t))$. As V increases, our method attempts to further reduce the energy consumption. That is, the service time allocated to each queue decreases as V increases. Thus, the average queue length increases, whereas the amount of energy consumed by a MES decreases with V. Therefore, by controlling V during the service-time allocation process, a MES operator can balance the energy consumption and queuing delay. In these figures, we also observe that, compared to WoPred, our method both consumes less energy and achieves a smaller average queue length for all Vs because our method considers the previous workload history reflected in $Q_i(t)$ and the workload expected during the current time slot $\hat{w}_i(t)$ when determining $\alpha_i(t)$s. In addition, the average energy consumption and average queue length obtained using our method are comparable to those achieved in an ideal case, in which $w_i(t)$ is known in advance.

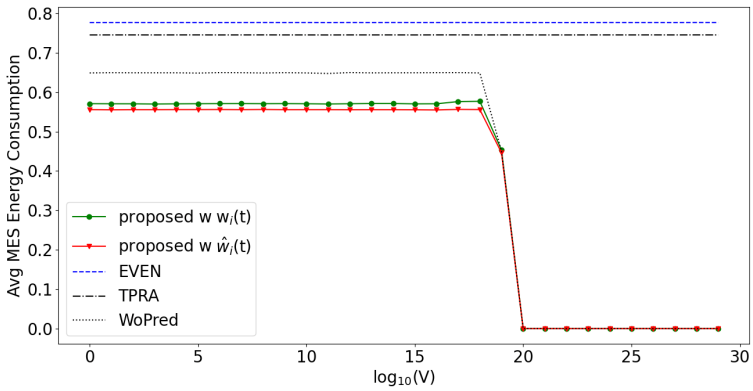
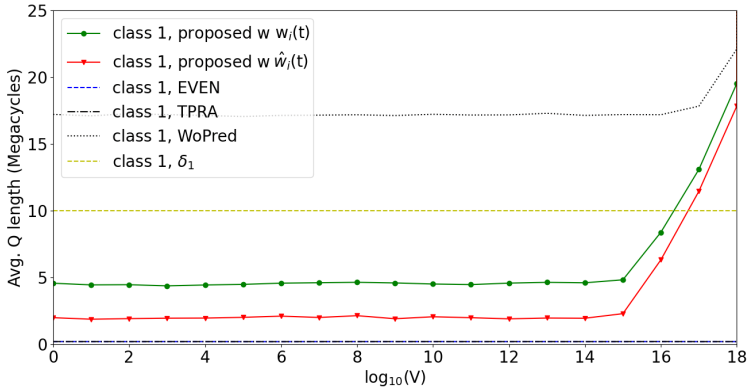
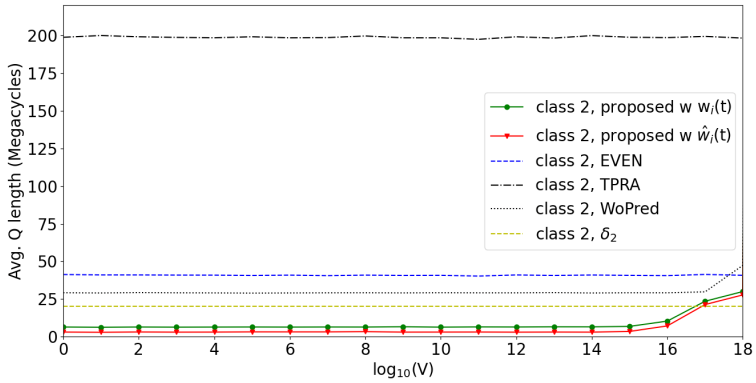


Figure 6. Average energy consumption of a MES with different V when $N = 3$.

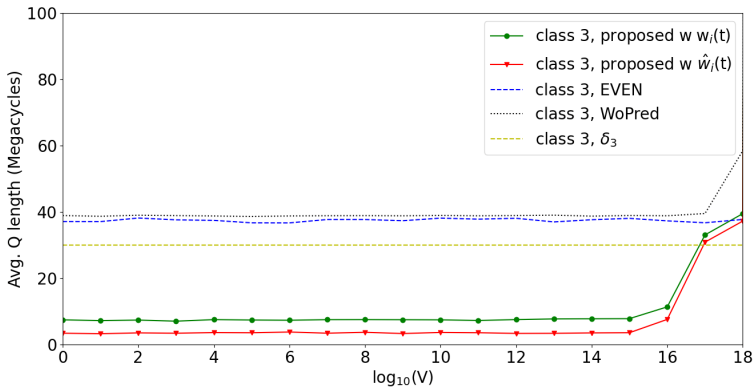


(**a**)

Figure 7. *Cont.*



(**b**)



(**c**)

Figure 7. Average queue length with different V. The average length of the Class 3 queue obtained by TPRA is more than 2100 megacycles, which is much larger than those achieved by other methods. Therefore, in Figure 7c, we omit the case when TPRA is applied to show the differences among the other methods visually. (**a**) Class 1, (**b**) Class 2, and (**c**) Class 3.

5.4. Scalability

In Figures 8 and 9, we show the average queue lengths of the different methods when the number of task classes (N) is 6 and 9, respectively. When $N = 6$, the average queue lengths obtained by TPRA are much larger than those achieved by other methods in Classes 4–6. For example, the average queue lengths obtained by TPRA are 11–25-times larger than those achieved by EVEN or WoPred. Moreover, in Class 6, the average queue length obtained by EVEN is much larger than those achieved by other methods. Therefore, in these subfigures, we omit the cases when TPRA and EVEN are applied to show the differences among the other methods visually. When $N = 9$, the average queue lengths obtained by TPRA are much larger than those achieved by other methods in Classes 7–9. Thus, in Classes 7–9, we omit the cases when TPRA is applied to show the differences among the other methods visually. In these figures, we also observe that the average queue lengths obtained by our method look identical to those obtained by the ideal method that knows $w_i(t)$s in advance. This is because they achieve similar average queue lengths for all Vs. For example, when $N = 6$ and $log_{10}V = 16$ (i.e., $V = 10^{16}$), the differences in the average queue length between our method and the ideal method ranges from 0.0053 to 0.097. When $N = 9$ and $V = 10^{16}$, the differences range from 0.0027 to 0.073. From Figures 8 and 9, we can observe that unlike the other methods, our method guarantees the average queue length below a given threshold for all the queues.

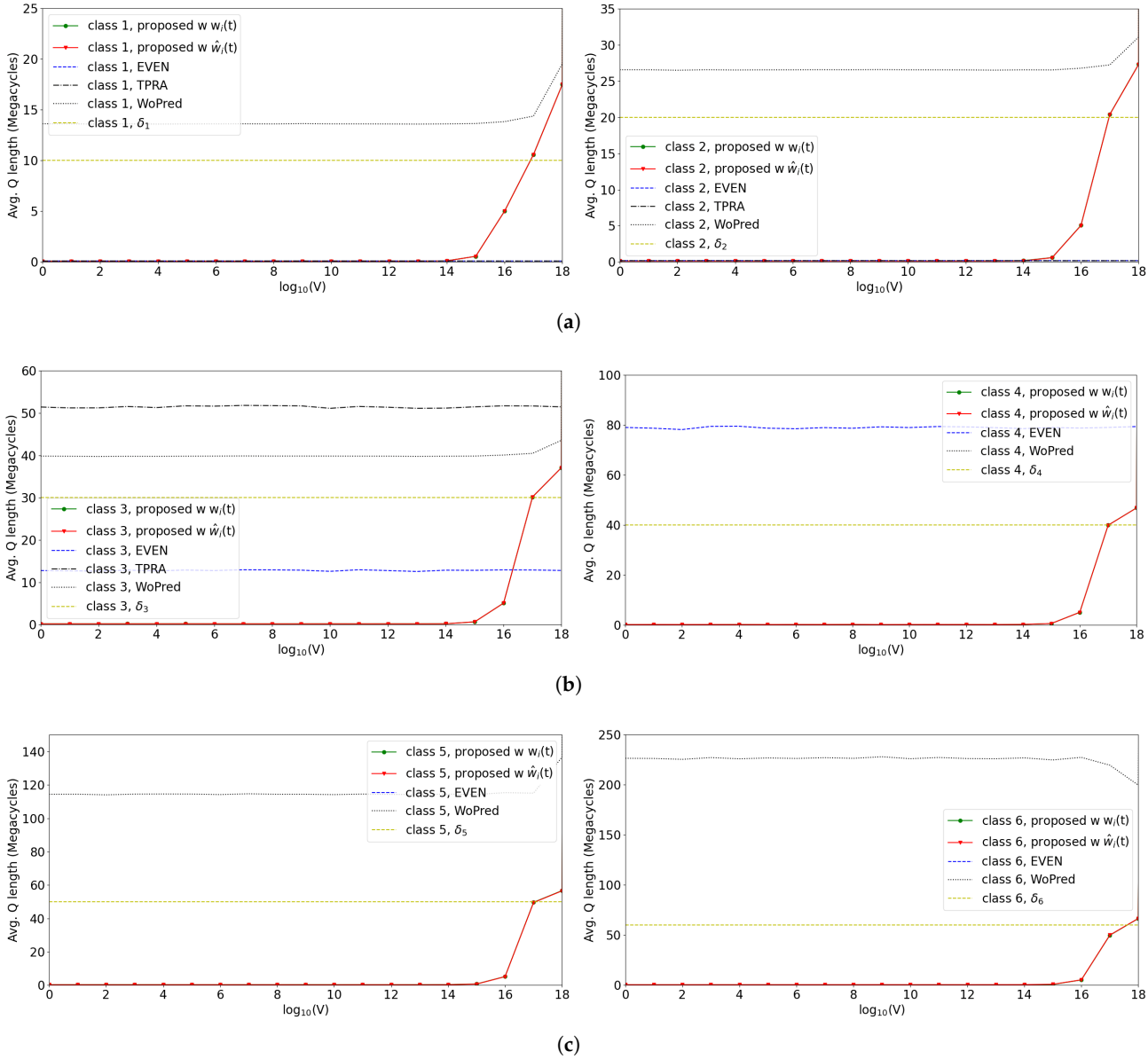


Figure 8. Average queue length with different Vs when the number of classes is 6. In Classes 4–6, the average queue lengths obtained by TPRA are much larger than those achieved by other methods. Therefore, in Classes 4–6, we omit the case when TPRA is applied and adjust the range of the x-axis to show the differences among the other methods visually. We also note that since the results obtained by the proposed method are almost identical to those obtained by the ideal case, they look the same in these figures. (**a**) Classes 1 and 2, (**b**) Classes 3 and 4, and (**c**) Classes 5 and 6.

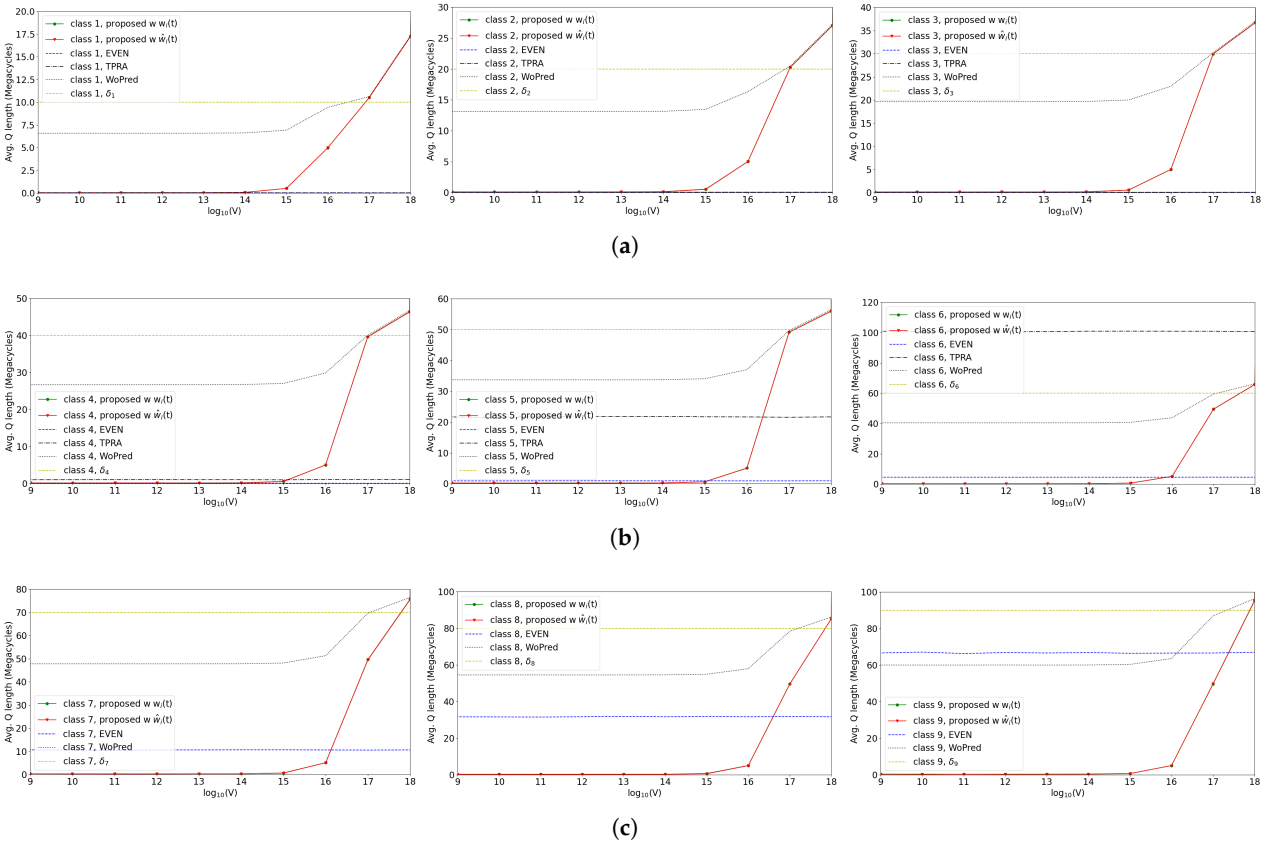


Figure 9. Average queue length with different Vs when the number of classes is 9. In Classes 7–9, the average queue lengths obtained by TPRA are much larger than those achieved by other methods. Therefore, in Classes 7–9, we omit the case when TPRA is applied to show the differences among the other methods visually. We also note that since the results obtained by the proposed method are almost identical to those obtained by the ideal case, they looks the same in these figures. (**a**) Classes 1, 2, and 3, (**b**) Classes 4, 5, and 6, and (**c**) Classes 7, 8, and 9.

In Figure 10, we show the average energy consumption of a MES when $N = 6$ (Figure 10a) and $N = 9$ (Figure 10b) with different Vs. We can see that the energy consumed by our method is comparable to that consumed by the ideal method. Moreover, we can observe that our method consumes the least energy. We can also observe that there is a trade-off between the average energy consumption and the average queue length, which we can control by using the parameter V.

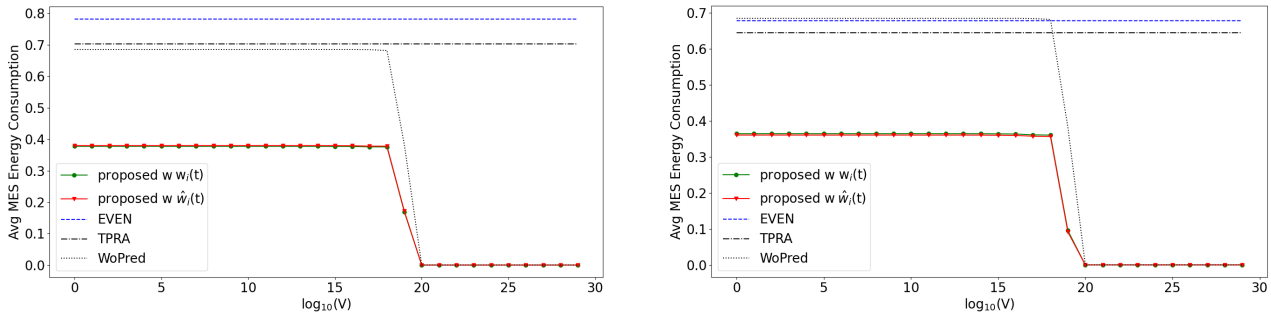


Figure 10. Average energy consumption of a MES with different Vs when $N = 6$ and $N = 9$. (**a**) $N = 6$ classes, and (**b**) $N = 9$ classes.

6. Conclusions and Future Works

In this study, we proposed an online service-time allocation method that can reduce the energy consumption of a MES while guaranteeing the queuing latency requirement for each class queue. We combined the Lyapunov optimization framework with a deep learning model and devised a lightweight algorithm that can quickly determine an optimal solution at the beginning of each time slot using the observed queue length and predicted workload. Using simulation studies, we verified that the proposed method is superior to conventional methods in reducing the energy consumption of a MES while maintaining the queuing delay below a given threshold value.

Since our method deals with the service-time allocation problem when a MES is not overloaded, it can complement a MES congestion resolution method and a task migration method. If a MES detects that the average queue length cannot be guaranteed because the task input rate to a MES exceeds the capacity of the MES, it can invoke a MES congestion resolution method to move some tasks to the other MESs or to a cloud server. In addition, when a MES detects that a device moves out of its service area, it invokes a task migration method to move tasks to other MESs.

As a future work, we plan to extend our resource management method into more complex scenarios where MESs migrate tasks and the backhaul network capacity is limited. To further improve the system performance and QoS requirements of tasks, we will jointly optimize a service-time allocation strategy, a MES congestion resolution policy, and a task migration algorithm.

Author Contributions: Conceptualization, J.P. and Y.L.; methodology, J.P.; software, Y.L.; formal. All authors have read and agreed to the published version of the manuscript.

Funding: The present research has been conducted by the Research Grant of Kwangwoon University in 2021. This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. 2021R1F1A1047113).

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

N	A set of task classes
f_{mes}	The CPU frequency of a MES
τ	The length of a time slot
$Q_i(t)$	The workload stored in queue i at the start of time slot t
$\alpha_i(t)$	The proportion of time that a MES serves a queue i during a time slot t
$b_i(t)$	The number of CPU cycles used to serve a queue i during a time slot t
$e_i(t)$	The amount of energy consumed by a MES to serve queue i during a time slot t
$E(t)$	The amount of energy consumed by a MES during time slot t
$w_i(t)$	The workload newly imposed on queue i during time slot t
$\bar{L}_i$	The long-term time-averaged queuing latency
δ_i	The queue length threshold of queue i
$Z_i(t)$	The virtual queue i during time slot t

References

1. Mao, Y.; You, C.; Zhang, J.; Huang, K.; Letaief, K.B. A Survey on Mobile Edge Computing: The Communication Perspective. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 2322–2358. [CrossRef]
2. Spinelli, F.; Mancuso, V. Toward Enabled Industrial Verticals in 5G: A Survey on MEC-Based Approaches to Provisioning and Flexibility. *IEEE Commun. Surv. Tutor.* **2021**, *23*, 596–630. [CrossRef]
3. Sarrigiannis, I.; Ramantas, K.; Kartsakli, E.; Mekikis, P.-V.; Antonopoulos, A.; Verikoukis, C. Online VNF Lifecycle Management in an MEC-Enabled 5G IoT Architecture. *IEEE Internet Things J.* **2020**, *7*, 4183–4194. [CrossRef]
4. Siriwardhana, Y.; Porambage, P.; Liyanage, M.; Ylianttila, M. A Survey on Mobile Augmented Reality with 5G Mobile Edge Computing: Architectures, Applications, and Technical Aspects. *IEEE Internet Things J.* **2021**, *23*, 1160–1192. [CrossRef]

5. Mach, P.; Becvar, Z. Mobile Edge Computing: A Survey on Architecture and Computation Offloading. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 1628–1656. [CrossRef]
6. Sana, M.; Merluzzi, M.; Pietro, N.d.; Strinati, E.C. Energy Efficient Edge Computing: When Lyapunov Meets Distributed Reinforcement Learning. In Proceedings of the IEEE International Conference on Communications Workshops (ICC Workshops), Montreal, QC, Canada, 14–23 June 2021.
7. Zhang, X.; Pal, A.; Debroy, S. Deep Reinforcement Learning Based Energy-efficient Task Offloading for Secondary Mobile Edge Systems. In Proceedings of the IEEE 45th LCN Symposium on Emerging Topics in Networking (LCN Symposium), Sydney, Australia, 16–19 November 2020.
8. Tu, Y.; Chen, H.; Yan, L.; Zhou, X. Task Offloading Based on LSTM Prediction and Deep Reinforcement Learning for Efficient Edge Computing in IoT. *Future Internet* **2022**, *14*, 30. [CrossRef]
9. Nath, S.; Wu, J. Dynamic Computation Offloading and Resource Allocation for Multi-user Mobile Edge Computing. In Proceedings of the IEEE Global Communications Conference (GLOBECOM), Taipei, Taiwan, 7–11 December 2020.
10. Wang, S.; Xu, J.; Zhang, N.; Liu, Y. A Survey on Service Migration in Mobile Edge Computing. *IEEE Access* **2018**, *6*, 23511–23528. [CrossRef]
11. Liu, C.; Tang, F.; Hu, Y.; Li, K.; Tang, Z.; Li, K. Distributed Task Migration Optimization in MEC by Extending Multi-Agent Deep Reinforcement Learning Approach. *IEEE Trans. Parallel Distrib. Syst.* **2021**, *32*, 1603–1614. [CrossRef]
12. Zhang, W.-Z.; Elgendy, I.A.; Hammad, M.; Iliyasu, A.M.; Du, X.; Guizani, M.; El-Latif, A.A.A. Secure and Optimized Load Balancing for Multitier IoT and Edge-Cloud Computing Systems. *IEEE Internet Things J.* **2021**, *8*, 8119–8132. [CrossRef]
13. Zhang, F.; Wang, M.M. Stochastic Congestion Game for Load Balancing in Mobile-Edge Computing. *IEEE Internet Things J.* **2021**, *8*, 778–790. [CrossRef]
14. Park, J.; Lim, Y. Balancing Loads among MEC Servers by Task Redirection to Enhance the Resource Efficiency of MEC Systems. *Appl. Sci.* **2021**, *11*, 7589. [CrossRef]
15. Huang, M.; Liu, W.; Wang, T.; Liu, A.; Zhang, S. A Cloud–MEC Collaborative Task Offloading Scheme with Service Orchestration. *IEEE Internet Things J.* **2020**, *7*, 5792–5805. [CrossRef]
16. Sohee, B.; Seungyul, H.; Youngchul, S. A Reinforcement Learning Formulation of the Lyapunov Optimization: Application to Edge Computing Systems with Queue Stability. *arXiv* **2020**, arXiv:2012.07279.
17. Porambage, P.; Okwuibe, J.; Liyanage, M.; Ylianttila, M.; Taleb, T. Survey on Multi-Access Edge Computing for Internet of Things Realization. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 2961–2991. [CrossRef]
18. 3GPP TS 23.203. Technical Specification Group Services and System Aspects; Policy and Charging Control Architecture, V17.2.0., 3GPP. 2021. Available online: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=810 (accessed on 15 February 2022).
19. IETF RFC 4594, Configuration Guidelines for DiffServ Service Classes, IETF. 2006. Available online: https://www.rfc-editor.org/info/rfc4594 (accessed on 15 February 2022).
20. Greff, K.; Srivastava, R.K.; Koutník, J.; Steunebrink, B.R.; Schmidhuber, J. LSTM: A Search Space Odyssey. *IEEE Trans. Neural Netw. Learn. Syst.* **2015**, *28*, 2222–2232. [CrossRef]
21. Schmidhuber, J.; Wierstra, D.; Gomez, F. Evolino: Hybrid Neuroevolution/Optimal Linear Search for Sequence Learning. In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), Edinburgh, UK, 30 July–5 August 2005.
22. Zhao, Z.; Chen, W.; Wu, X.; Chen, P.C.Y.; Liu, J. LSTM Network: A Deep Learning Approach for Short-term Traffic Forecast. *IET Intell. Transp. Syst.* **2017**, *11*, 68–75. [CrossRef]
23. Boyd, S.; Vandenberghe, L. *Convex Optimization*; Cambridge University Press: Cambridge, UK, 2004.
24. Ma, Y.; Liang, W.; Huang, M.; Xu, W.; Guo, S. Virtual Network Function Service Provisioning in MEC via Trading Off the Usages between Computing and Communication Resources. *IEEE Trans. Cloud Comput.* **2020**, *2020*, 1–15. [CrossRef]
25. Huang, X.; He, L.; Wang, L.; Li, F. Towards 5G: Joint Optimization of Video Segment Caching, Transcoding and Resource Allocation for Adaptive Video Streaming in a Multi-Access Edge Computing Network. *IEEE Trans. Veh. Technol.* **2021**, *70*, 10909–10924. [CrossRef]
26. Guo, C.; He, W.; Li, G.Y. Optimal Fairness-Aware Resource Supply and Demand Management for Mobile Edge Computing. *IEEE Wirel. Commun. Lett.* **2021**, *10*, 678–682. [CrossRef]
27. Liu, B.; Liu, C.; Peng, M. Resource Allocation for Energy-Efficient MEC in NOMA-Enabled Massive IoT Networks. *IEEE J. Sel. Areas Commun.* **2021**, *39*, 1015–1027. [CrossRef]
28. Feng, L.; Li, W.; Lin, Y.; Zhu, L.; Guo, S.; Zhen, Z. Joint Computation Offloading and URLLC Resource Allocation for Collaborative MEC Assisted Cellular-V2X Networks. *IEEE Access* **2020**, *8*, 24914–24926. [CrossRef]
29. Wang, K.; Ding, Z.; So, D.K.C.; Karagiannidis, G.K. Stackelberg Game of Energy Consumption and Latency in MEC Systems with NOMA. *IEEE Trans. Commun.* **2021**, *69*, 2191–2206. [CrossRef]
30. Yang, X.; Luo, H.; Sun, Y.; Zou, J.; Guizani, M. Coalitional Game-Based Cooperative Computation Offloading in MEC for Reusable Tasks. *IEEE Internet Things J.* **2021**, *8*, 12968–12982. [CrossRef]
31. Moura, J.; Hutchison, D. Game Theory for Multi-Access Edge Computing: Survey, Use Cases, and Future Trends. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 260–288. [CrossRef]

32. Zhang, Y.; Zhou, X.; Teng, Y.; Fang, J.; Zheng, W. Resource Allocation for Multi-User MEC System: Machine Learning Approaches. In Proceedings of the International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA, 12–14 December 2018.
33. Peng, H.; Shen, X.S. DDPG-based Resource Management for MEC/UAV-Assisted Vehicular Networks. In Proceedings of the IEEE Vehicular Technology Conference (VTC), Victoria, BC, Canada, 18 November–16 December 2020.
34. Chou, P.-Y.; Chen, W.-Y.; Wang, C.-Y.; Hwang, R.-H.; Chen, W.-T. Deep Reinforcement Learning for MEC Streaming with Joint User Association and Resource Management. In Proceedings of the IEEE International Conference on Communications (ICC), Dublin, Ireland, 7–11 June 2020.
35. Matheron, G.; Perrin, N.; Sigaud, O. The Problem with DDPG: Understanding Failures in Deterministic Environments with Sparse Rewards. *arXiv* **2019**, arXiv:1911.11679.
36. Zhang, Q.; Gui, L.; Hou, F.; Chen, J.; Zhu, S.; Tian, F. Dynamic Task Offloading and Resource Allocation for Mobile-Edge Computing in Dense Cloud RAN. *IEEE Internet Things J.* **2020**, *7*, 3282–3299. [CrossRef]
37. Vijayashree, K.V.; Anjuka, A. Fluid Queue Driven by an Queue Subject to Bernoulli-Schedule-Controlled Vacation and Vacation Interruption. *Hindawi Adv. Oper. Res.* **2016**, *2016*, 1–11. [CrossRef]
38. Burd, T.D.; Brodersen, R.W. Processor Design for Portable Systems. *J. Vlsi Signal Process. Syst. Signal Image Video Technol.* **1996**, *13*, 203–221. [CrossRef]
39. Neely, M.J. *Stochastic Network Optimization with Application to Communication and Queuing Systems*; Morgan & Claypool: San Rafael, CA, USA, 2010.
40. Telecom Italia. Telecommunications—SMS, Call, Internet—MI, Havard Dataverse. 2015. Available online: https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/EGZHFV (accessed on 10 January 2022)
41. Qiu, X.; Zhang, W.; Chen, W.; Zheng, Z. Distributed and Collective Deep Reinforcement Learning for Computation Offloading: A Practical Perspective. *IEEE Trans. Parallel Distrib. Syst.* **2021**, *32*, 1085–1101. [CrossRef]
42. Wang, S.; Zhang, X.; Yan, Z.; Wang, W. Cooperative Edge Computing with Sleep Control under Nonuniform Traffic in Mobile Edge Networks. *IEEE Internet Things J.* **2019**, *6*, 4295–4306. [CrossRef]