applied sciences

MDPI

Article

Bio-Inspired Sleep Control for Improving the Energy Efficiency of a MEC System

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

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

Abstract: The energy consumption of a multi-access edge computing (MEC) system must be reduced to save operational costs. Determining a set of active MEC servers (MECSs) that can minimize the energy consumption of the MEC system while satisfying the service delay requirements of the tasks is an NP-complete problem. To solve this problem, we take a bio-inspired approach. We note that the sleep control problem of the MECS differentiates the operational mode among neighboring MECSs. Therefore, by mimicking the cell differentiation process in a biological system, we designed a distributed sleep control method. Each MECS periodically gathers the utilization and delta levels of the neighboring MECSs. Subsequently, by using the gathered information and the Delta–Notch inter-cell signaling model, a MECS autonomously decides whether to sleep. We evaluated the performance of our method through extensive simulations. Compared with a conventional method, the proposed method reduces energy consumption in a MEC system by more than 13% while providing a comparable service delay. In addition, our method reduces the variations in the service delay by more than 35%.

Keywords: MEC energy saving; MECS sleep control; Task QoS; inter-cell signaling model; bio-inspired approach

Citation: Park, J.; Lim, Y. Bio-Inspired Sleep Control for Improving the Energy Efficiency of a MEC System. *Appl. Sci.* **2023**, *13*, 2620. https://doi.org/10.3390/app13042620

Academic Editors: Teen-Hang Meen, Charles Tijus, Po-Lei Lee and Chun-Yen Chang

Received: 21 December 2022
Revised: 14 February 2023
Accepted: 16 February 2023
Published: 17 February 2023

1. Introduction

Mobile applications running on resource-constrained mobile devices require high computing power and low latency. To satisfy these requirements, multi-access edge computing (MEC) systems have drawn much attention from industry and academia [1–3]. In a MEC system, multiple MEC servers (MECSs) are deployed at the edges of the access networks. Mobile devices offload their computationally intensive tasks to these MECSs that process these tasks and return the results to the mobile devices. By providing high computing capacities near mobile devices, a MEC system is expected to facilitate computing-intensive and delay-stringent mobile applications. In general, the service area of a MECS is much smaller than that of a conventional cloud server. Additionally, the capacity of a MECS is lower than that of a cloud server. Therefore, an increasing number of MECSs is required with the increase in the service region of a MEC system. As a result, the operational cost of the MEC system increases. In addition, networks and data centers are expected to account for 59.8% of the CO_2 emissions in the information and communications technology sector by 2030 [4]. Therefore, reducing the energy consumption of a MEC system is important for reducing the operational cost of a MEC system and tackling global warming.

A MECS consumes considerable energy even when it is in an idle state [5]. The total energy consumption of a MEC system can be decreased if an idle MECS enters the sleep state. However, because the loads imposed on a MEC system are processed only by active MECSs, the service delay owing to the MEC system increases with the increase in the number of MECSs in the sleep state. Therefore, an appropriate sleep control method for MECS that can reduce the total energy consumption of a MEC system while providing reasonable service delay is required.

The sleep control problem in a MEC system is a combinatorial problem that determines the set of MECSs that must be in the sleep state for a given task distribution in a MEC system. Optimization-based methods have been proposed to address the energy-saving issue in MEC systems [5–7]. These methods formulate an optimization problem and transform the original problem into a set of solvable mathematical forms that can be used to design an algorithm to solve the problem. However, the complexities of optimization-based algorithms increase exponentially with the increase in the number of MECS. In addition, rapidly adjusting to the dynamics of a MEC system is difficult because these algorithms are based on a network snapshot. Therefore, to ensure the scalability and adaptability of a MEC system, a lightweight distributed approach is required to reduce energy consumption while satisfying the service delay requirement. To achieve this objective, we adopted a bio-inspired model and devised a method for a MECS to make an automatic sleep decision. Bio-inspired approaches have drawn considerable attention in the communications and networking domains for resolving the scalability issue in an adaptive and distributed manner [8,9]. Bio-inspired modeling techniques exploit the inherent adaptability of biological systems by mimicking the behavior of objects in such systems to solve engineering problems.

We regarded the MECS sleep decision process as a state differentiation process among the MECSs. More specifically, each MECS differentiates the operational state, i.e., sleep or active, from the neighboring MECSs by considering the relative load to those of the neighbors. We subsequently designed a sleep decision process using the Delta–Notch lateral inhibition model that explains the process by which the fate of an endothelial cell is determined during sprouting angiogenesis using the delta protein level of the cell relative to those of its surrounding cells. Additionally, we devised a MECS activation scheme. The load on a MECS varies with time and space [10]. As the number of tasks offloaded to an active MECS increases, the MECS is likely to violate the service delay requirement. To avoid this situation, we enabled an active MECS to wake up neighboring MECSs that are sleeping before the service time exceeds a predetermined threshold value. Through extensive simulations, we analyzed the performance of the proposed method in terms of energy consumption and the queuing delay of a MEC system. We demonstrated that the proposed strategy can reduce the energy consumption of a MEC system while maintaining the length of the queue below a predetermined threshold even under different system workloads, the number of MECSs, and the ranges of neighborhoods of these MECSs. We summarize our main contributions as follows.

- We address the energy-saving problem in a MEC system. To resolve the problem, we propose a MECS sleep control method by adopting a bio-inspired approach. By mimicking the cell differentiation process in a biological system, our method enables each MECS to make a sleep decision in a distributed manner.
- We analyze our sleep control method, which is composed of a sleep decision module, a task transfer module, and a MECS activation module. We show that the worst-case time complexity of our sleep control method is $O(|N_i|)$. Therefore, our method scales linearly with the number of neighboring MECSs.
- We compare the performance of our method with that of a conventional method through simulation studies. We show that compared with a conventional method, our method can reduce the energy consumption of a MEC system while providing a comparable task service delay. We also show that our method can provide a more stable service delay by reducing the variation in the length of the task queue in a MECS.

The remainder of this paper is organized as follows. In Section 2, we discuss energy-saving issues in a MEC system. We explain the system model in Section 3 and present our bio-inspired MECS sleep-control strategy in Section 4. In Section 5, we verify our method through simulations. We conclude the study by discussing future research directions in Section 6. Before we proceed, in Table 1, we present the notations used in this paper.

Table 1. Notations used.

Notations	Meaning
N_i	A set of neighboring MECSs of a MECS i
b_i	Task service rate of a MECS i
$Q_i(t)$	The length of a task queue in a MECS i at the beginning of a time slot t
$a_i(t)$	The number of tasks received by a MECS i during a time slot t
$e_i(t)$	The amount of energy consumed by a MECS i during a time slot t
$\mu_i(t)$	Utilization of a MECS i during a time slot t.
$\beta_i(t)$	Operation mode of a MECS i during a time slot t
$d_i(t)$	Delta activation level of a MECS i during a time slot t
$n_i(t)$	Notch activation level of a MECS i during a time slot t
$f(x)$	A function determining the production rate of the Notch activation level
$g(x)$	A function determining the production rate of the Delta activation level
$\eta_{i,j}(t)$	Proportion of tasks transferred from a sleeping MECS i to its neighbor j

2. Related Works

2.1. Resource Management in a MEC System

To satisfy the quality of service requirements of the tasks offloaded to a MEC system in a resource-efficient manner, various methods have been proposed for diverse problems. The authors in [11] investigate a DNN model splitting problem in a hybrid mobile edge environment. They formulate a utility maximization problem with SLA, priority, and fairness constraints. After transforming the optimization problem into the multi-choice knapsack problem, they propose a feasible algorithm called HiTDL to solve the transformed problem. HiTDL achieves an optimal system throughput by making an informed decision on DNN model splitting and resource allocation while achieving SLA, priority, and fairness goals.

In [12], the demand response problem in edge cloud is explored. After deriving the operational costs of the cloudlet operator and the service providers, the authors formulate a long-term social cost minimization problem. Then, they propose an online auction algorithm called EdgeDR for performing the power emergency demand response and the computing demand response at the edge cloud.

The network selection and service placement problem are inspected in [13]. By incorporating the queuing delay, switching cost, and communication delay, an offline network selection and service placement optimization problem is formulated. The authors decompose the long-term optimization problem into a series of one-shot subproblems. A two-phase algorithm based on the matching game and the coalitional game is proposed to solve the subproblems. The proposed algorithm improves the QoS by balancing the queuing delay, communication delay, and service switching cost.

The authors in [14] tackle the multi-task transfer learning problem on the edge. They propose a task importance metric that measures the contribution of a task to the improvement of the decision-making performance. Then, they formulate the task allocation problem for the multi-task transfer learning on the edge by considering the task importance, the execution time, and resource limitations. To solve the formulated problem, a data-driven cooperative task allocation (DCTA) method is proposed by using reinforcement learning. DCTA reduces the processing time and saves energy consumption.

In [15], the authors probe into the container caching problem in a MEC system. By considering the startup delay of a container, service latency cost, and container retention cost, they formulate a system cost minimization problem as a linear integer problem. To resolve the problem, they propose an online algorithm named O-RDC. To reduce the system cost, O-RDC opportunistically distributes requests by considering the resource capacities and network delays between edge nodes.

An admission control policy for time-sensitive services is investigated in [16]. To maximize the revenue for the service provider while guaranteeing QoS for the accepted service requests, a service throughput maximization problem is formulated. A threshold structure is carefully defined and an optimal admission control mechanism called OA2 is devised

by taking the advantage of the strategic queuing approach. Through simulation studies using both synthetic traces and real-world service request traces, OA2 is verified to achieve maximal revenue while providing QoS guarantee in expectation for the accepted requests.

In [17], a fine-grained warm water cooling system is proposed to relieve the multiple-level hotspots and save the cooling energy of high-density and heterogeneous edge data centers. The authors remove the hardware-level hotspots by well designing a water circulation method. They also disperse the chip-level hotspots without manual intervention by developing new cold plates with vapor chambers.

The authors in [18] scrutinize the VNF chain deployment problem with the latency guarantee and the resource efficiency. They formulate the problem as a mixed integer linear programming problem. To resolve the formulated problem, they devise a two-stage latency-aware VNF deployment method. In the first stage, paths are selected and VNFs are assigned in the second stage. By considering the VNF instance reuse and the latency requirements, the proposed method jointly optimizes the computing resources of the edge servers and the bandwidth resources of the physical links.

In [19], a cost-efficient edge resource management platform for the NFV network called Finedge is proposed. To devise Finedge, the authors derived useful insights into the influence of the network function's flow-level characteristics and CPU allocation on the performances through empirical experiments. Using these insights, they designed Finedge, which can automatically assign the most suitable CPU core and the most cost-efficient CPU quota for each network function by taking its flow-level characteristics and QoS requirements into account.

2.2. Energy Saving in a MEC System

Various technical approaches have been adopted to reduce the energy consumption of MEC systems. The authors of [5] proposed a method that minimizes the long-term energy consumption of a backhaul network connecting MECSs. To achieve this, they adopted the Lyapunov-based approach and periodically decided whether to migrate tasks from one MECS to another. The Lyapunov optimization approach was also adopted in [7] to incorporate a MEC system into dense cellular networks. However, the authors considered only the task offloading between MECSs and a cloud server when they made joint computation offloading and MECS sleep decisions.

Deep reinforcement learning (DRL)-based methods were developed in [20,21]. The authors in [20] formulated an optimization problem that minimizes the energy consumption required to process all tasks imposed on a MEC system while satisfying the delay constraints of each UE. They presented conventional decomposition and deep Q-learning-based methods that solved the formulated problem by determining the offloading ratio of each UE and the optimal frequency of a central processing unit that a MECS should allocate to each UE. However, they considered only a simple environment, in which multiple UEs offload their tasks to a single MECS. In [21], the authors considered a NOMA-based MEC system that allows the use of computational resources of idle devices through machine-to-machine (M2M) task-offloading. They formulated an optimization problem for minimizing the energy consumption of the system under task-delay constraints and proposed a DRL-based sleep scheduling method for M2M devices.

In [22], the authors investigated the effectiveness of a queuing model compatible with the dynamics of traffic flow over a long-time scale. They employed the $M^x/D/N$ queuing system and derived the minimum number of active processing units that can guarantee the desired upper bound on the average queuing delay. Through simulation studies using real-world traffic traces, they demonstrated that the queuing model underestimates the delay and imparts aggressiveness to the sleeping strategy with respect to energy consumption.

The authors in [23] defined network cost by determining operational cost and a response delay. They subsequently proposed path selection and sleep scheduling methods by adopting a game-theoretic approach to minimize the network cost. However, acquiring the information needed to determine the path from an AP to a MECS is difficult. These

include the aggregate load on each router and each MECS in the system and the edges in the selected path. In addition, multiple iterations are required between the central coordinator and all APs. In [24], minority game theory was used to devise a server activation scheme that can support the requirements associated with the quality of experience of users in an energy-efficient manner. However, the authors assumed that the task arrival pattern follows a Poisson distribution, which may not reflect real-world scenarios.

In [25], the Bayesian learning automatan was used for a MEC server to make autonomous sleep decisions. The reputation of a MECS is developed according to the service quality evaluation presented by the users. However, a MECS is required to obtain feedback from the MEC environment regarding its decisions. More specifically, global information should be shared among the MEC servers to make sleep decisions. Additionally, their method requires a human-based evaluation function that is subjective for the computing system. Furthermore, MEC servers are assumed to be synchronized in terms of decision time and information shared among them.

In [6], the authors formulated a joint optimization problem to minimize the energy consumption of mobile-edge computing and caching (MECC)-enabled software-defined mobile networks. To resolve the problem, they decomposed the original problem into content source selection and bandwidth provisioning problems using dual decomposition. Subsequently, they solved the subproblems using the alternating direction method of multipliers in a distributed manner. In [26], a multi-state stochastic problem was formulated to model the energy-aware application placement problem in a MEC system. A parallel sample average approximation (SAA) algorithm was proposed to solve the formulated problem. Because SAA approximates the recourse function based on the sample average, SAA experiences a tradeoff between the complexity and accuracy of the algorithm.

In this study, we devised a MECS sleep control method that can reduce the total energy consumption of a MEC system. We followed a bio-inspired approach and devised the lightweight microscopic behavior of each MECS that can result in a globally desirable system property. This property is the energy saving of a MEC system that can be achieved if each MECS repeats the microscopic behavior using the local information obtained by interactions with only the local neighboring MECSs.

2.3. Bio-Inspired Networking

Bio-inspired methods exhibit inherent adaptability in nature and exploit inherent features to design a scalable, robust, and adaptive system that is capable of simple local interactions between the entities comprising the system. Biological design approaches have been applied to various fields in the communication and networking domains, including MAC design, resource allocation, routing, and security [27–29]. These bio-inspired methods demonstrated that the system reaches a globally optimal state, in which a system-wide optimization objective is accomplished even though each node comprising the large complex system repeatedly performs a simple operation with limited local information.

Among the bio-inspired models, we adopted the Delta–Notch inter-cell signaling model to design a MECS sleep control scheme. The Delta–Notch lateral inhibition process determines the fate of endothelial cells during sprouting angiogenesis using the levels of Delta and Notch proteins in a cell. The Delta and Notch levels of a cell are regulated in response to the average Delta level of neighboring cells. This lateral inhibition process is modeled using the following coupled ordinary differential equations [30,31]:

$$\frac{\partial n_i}{\partial t} = f(\bar{d}_i) - n_i \tag{1}$$

$$\frac{\partial d_i}{\partial t} = \nu(g(n_i) - d_i), \tag{2}$$

where n_i is the Notch level in cell i and d_i is the Delta level. Additionally, $\bar{d}_i$ is the average Delta level surrounding cell i and ν is a positive constant. Functions f and g represent the production rates of the Notch and Delta levels, respectively. These are expressed as

$$f(x) = \frac{x^k}{a + x^k}$$
$$g(x) = \frac{1}{1 + bx^h},$$

where $a, b > 0$ and $k, h \geq 1$.

3. System Model

We consider a MEC system composed of a set N of MECSs. A MECS j is said to be a neighbor of a MECS i if they exchange their local information. Each MECS i is configured to have a set N_i of neighboring MECSs. The neighboring MECSs are connected by a backhaul network. we denote the communication bandwidth between a MECS i and the neighboring MECS $j \in N_i$ as $b_{i,j}$. The control time is divided into discrete time slots of length τ. Each MECS possesses a task queue to accommodate the tasks it receives. There can be different types of tasks and they can have different QoS requirements. A MECS can accommodate them by maintaining a separate task queue for each task type. Because multiple queues share the computational resources of a MECS, optimally scheduling them affects the resource efficiency and energy consumption of a MECS. The task service time scheduling problem in a MECS is investigated in [10]. Thus, in the work, we assume that tasks are homogeneous.

We denote the length of the task queue in a MECS i at the beginning of a time slot t as $Q_i(t)$. We also denote the task service rate of a MECS i as b_i. The number of tasks arriving at a MECS i during a time slot t is denoted by $a_i(t)$. Since the stochastic characteristic of $a_i(t)$ is not known and varies in time and space, we do not make any assumption about the task arrival process. Instead, we measure $a_i(t)$ at each time slot and devise a sleep control mechanism by using the measured $a_i(t)$s. $a_i(t)$ can be subdivided into two groups. The first group is composed of the tasks offloaded directly from the devices to a MECS. We denote the number of tasks in the first group as $a_{1,i}(t)$. The second group of tasks is composed of the tasks migrated from the neighboring MECSs that decide to sleep. We denote the number of tasks in the second group by $a_{2,i}(t)$. Then, $a_i(t) = a_{1,i}(t) + a_{2,i}(t)$. We assume a MECS serves the tasks in its tasks queue in a FIFO manner. Since the control time is divided into a time slot, tasks are migrated at the end of a time slot. Thus, the tasks in the second group are stored after the tasks in the first group.

3.1. Service Delay

If tasks are offloaded to a MECS i during a time slot t and they are served by i, the service delays of these tasks are less than $s_{lo}(t) = Q_i(t-1) + a_{1,i}(t)/b_i$. When a MECS i decides to sleep at the end of a time slot, it must transfer the tasks in its task queue to the neighboring MECSs that are active. If we denote the proportion of tasks transferred from a MECS i to a MECS $j \in N_i$ as $\eta_{i,j}$, the communication delay between i and j becomes $d_{i,j}(t) = \eta_{i,j} Q_i(t)/b_{i,j}$. Then, the maximum service delay of the tasks transferred from a MECS i to a MECS j during a time slot t becomes $s_{tr}(t) = \eta_{i,j} Q_i(t)/b_{i,j} + (Q_j(t) + a_j(t))/b_j$.

3.2. MECS Energy Consumption

In [23], measurements have revealed that the power consumption of a server is composed of idle power and active power that varies according to the workload of the server. Based on these results, an energy consumption model was proposed in [5]. By following this load-dependent energy consumption model, we express the energy consumed by an active MECS as

$$e_i^p(t) = \alpha P_m + (1 - \alpha) P_m \mu_i(t), \tag{3}$$

where P_m is the maximum power that a MECS i consumes when it is fully utilized, α is the proportion of the power that determines the energy consumption of a MECS i in an idle state, and $\mu_i(t) = \min\{Q_i(t) + a_i(t), \tau b_i\}/\tau b_i$ is the utilization of a MECS i during a time slot t.

When a MECS i decides to sleep, it has to distribute the tasks in $Q_i(t)$ to its neighbors. If we denote by ε_{tr} the energy consumed to transfer a task from a MECS to its neighbor, the energy consumed by a MECS i to transfer the tasks in its task queue becomes

$$e_i^{tr}(t) = \varepsilon_{tr} Q_i(t). \tag{4}$$

Then, the energy consumed by a MECS i during a time slot t is given as

$$e_i(t) = \beta_i(t) e_i^p(t) + (1 - \beta_i(t)) e_i^{tr}(t), \tag{5}$$

where $\beta_i(t)$ is an indicator function that represents the operational mode of a MECS i during a time slot t. More specifically, $\beta_i(t) = 1$ indicates that a MECS i is in an active state during a time slot t. By contrast, $\beta_i(t) = 0$ indicates that a MECS i is in a sleep state during a time slot t.

3.3. Problem Formulation

A MEC system must provide a reasonable service level in terms of the task service delay. Specifically, we aim to restrain the maximum task service delay of the tasks accommodated by a MEC system in each time slot below a given threshold. Therefore, our sleep control problem related to a MECS involves determining an optimal sleep control vector $\vec{\beta}^*(t) = \{\beta_1^*(t), \ldots, \beta_{|N|}^*(t)\}$ while maintaining $\max\{s_{lo}(t), s_{tr}(t)\}$ below a predetermined threshold value s_{th}. Thus, the MECS sleep control problem can be expressed as follows:

$$\begin{aligned} \vec{\beta}^*(t) = \arg\min_{\vec{\beta}(t)} & \sum_{i \in N} \beta_i(t) e_i^p(t) + (1 - \beta_i(t)) e_i^{tr}(t) \\ s.t. \quad & \max\{s_{lo}(t), s_{tr}(t)\} \leq s_{th}, \quad \forall i \in N, \forall t. \end{aligned} \tag{6}$$

The optimization problem in Equation (6) is a binary integer programming problem, which is one of Karf's NP-complete problems. Therefore, in this study, we adopt a bio-inspired approach and devise a lightweight heuristic algorithm that can resolve this problem in a distributed manner.

3.4. Overview of Sleep Control Mechanism

A MECS sleep control method has to decide both when to sleep and when to wake up sleeping MECSs. When a MECS i decides to sleep, a sleep control method has to determine the way to process the tasks in Q_i. When a MECS i decides to wake up sleeping neighbors, it has to determine the set of neighbors to be woken up and the way to distribute the tasks in Q_i to the woken-up neighbors.

In Figure 1, we illustrate an overview of our sleep control mechanism. Figure 1a shows a sleep decision procedure. Each MECS i periodically exchanges its utilization $(\mu_i(t))$ and the Delta activation level $(d_i(t))$ with its active neighboring MECSs j and k. By incorporating the relative utilization and the relative Delta activation level in the Delta–Notch inter-cell signaling model, a MECS i determines its sleep probability. The sleep probability increases as the relative utilization of a MECS becomes smaller. Therefore, it is likely that a relatively underutilized MECS decides to sleep. When a MECS i decides to sleep (i.e., $\beta_i = 0$), the tasks in Q_i are transferred to all its active MECSs j and k in inverse proportion to their utilizations to avoid the situation where an active MECS becomes overloaded by accommodating the tasks transferred from the MECSs that decide to sleep.

In Figure 1b, we depict a wake-up procedure. The task service delay increases with the length of the task queue in a MECS. Thus, when an active MECS i detects that Q_i is larger than a predefined threshold, it wakes up all its sleeping neighbors j and k and distributes a fraction of the tasks in its task queue to them to decrease its service delay.

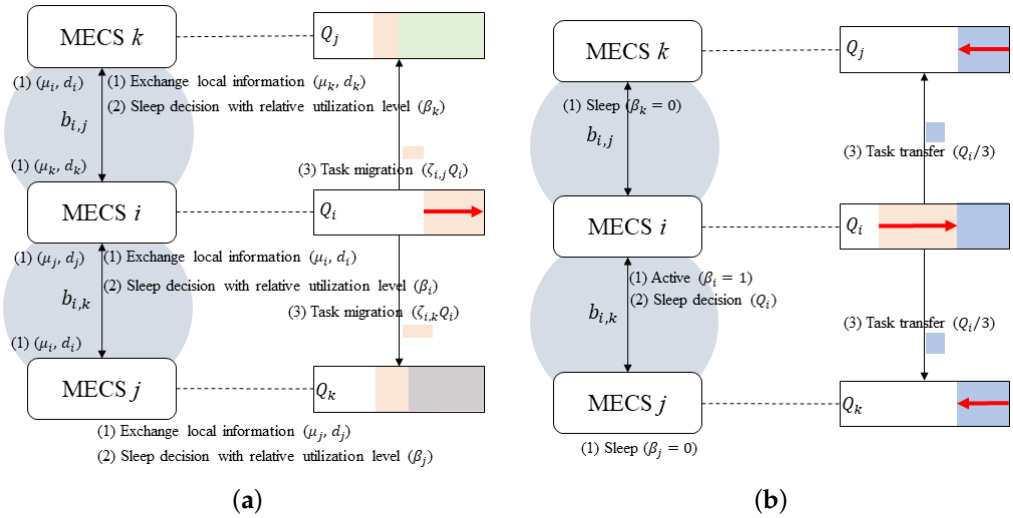


Figure 1. An overview of the sleep control mechanism. (**a**) Sleep example; (**b**) Wake-up example.

4. Bio-Inspired MECS Sleep Control Method

4.1. Sleep Decision by a MECS

We use the lateral inhibition model to enable each MECS to autonomously determine its operational mode during a time slot. Lateral inhibition is a differentiation process among neighboring cells using Delta–Notch activation levels. We regard the sleep decision problem as a differentiation process because a MECS attempts to differentiate the operating mode (sleep or active) from the neighboring MECSs according to the utilization level. The Delta activation level of a cell represents the amount of inhibitory power exerted on neighboring cells. Therefore, we use the Delta activation level as the sleep probability of a MECS by ensuring that it is inversely proportional to the relative utilization to the local average utilization of the neighboring MECSs. We present the sleep decision algorithm of a MECS in Algorithm 1.

The Notch activation level of a MECS i at the beginning of a time slot t is denoted as $n_i(t)$. Additionally, we denote the Delta activation level of MECS i at the beginning of time slot t as $d_i(t)$. At the beginning of each time slot, each MECS i calculates its utilization $\mu_i(t) = \frac{\min\{Q_i(t)+\tau a_i(t), \tau b_i\}}{\tau b_i}$ and exchanges $\mu_i(t)$ and $d_i(t)$ with the neighboring MECSs ($\forall j \in N_i$) and calculates the average local utilization ($\bar{\mu}_i(t)$) and average local Delta level ($\bar{d}_i(t)$).

$$\bar{\mu}_i(t) = \frac{1}{|N_i|} \sum_{j \in N_i} \mu_j(t). \tag{7}$$

$$\bar{d}_i(t) = \frac{1}{|N_i|} \sum_{j \in N_i} d_j(t). \tag{8}$$

Subsequently, a MECS i updates $n_i(t)$ and $d_i(t)$ K times or until convergence, whichever comes first (Lines 7–11). That is, for each iteration k, the Delta and Notch activation levels ($d'_i(k)$ and $n'_i(k)$, respectively) are updated as follows:

$$n'_i(k+1) = (1 - l_n) n'_i(k) + l_n f(\bar{d}_i(t)),$$
$$d'_i(k+1) = (1 - l_d) d'_i(k) + l_d g\left(n'_i(k+1) \frac{\mu_i(t)}{\bar{\mu}_i(t)} \right)$$

where l_n and l_d are the learning rates of the Notch and Delta activation levels, respectively. According to [30], the functions $f(\cdot)$ and $g(\cdot)$ are given by

$$f(x) = \frac{x^k}{a + x^k}$$
$$g(x) = \frac{1}{1 + bx^h},$$

where $a = 0.01, b = 100$, and $k = h = 2$.

Algorithm 1 Bio-Inspired Sleep Decision Algorithm

1: **At the beginning of time slot *t*:**
2: MECS i collects $\mu_j(t)$s and $d_j(t)$s from all $j \in N_i$.
3: $\bar{\mu}_i(t) = \frac{1}{|N_i|} \sum_{j \in N_i} \mu_j(t)$.
4: $\bar{d}_i(t) = \frac{1}{|N_i|} \sum_{j \in N_i} d_j(t)$.
5: $k = 0, \epsilon_d = 10^{-4}$.
6: $n'_i(k) = n_i(t), d'_i(k) = d_i(t), \delta_d = \epsilon_d + 1$.
7: **while** $k < K \;||\; \delta_d > \epsilon_d$ **do**
8: $n'_i(k+1) = (1 - l_n) n'_i(k) + l_n f(\bar{d}_i(t))$.
9: $d'_i(k+1) = (1 - l_d) d'_i(k) + l_d g\left(n'_i(k+1) \frac{\mu_i(t)}{\bar{\mu}_i(t)}\right)$.
10: $\delta_d = d'_i(k+1) - d'_i(k)$
11: $k++$.
12: $n_i(t) = n'_i(k-1)$.
13: $d_i(t) = d'_i(k-1)$.
14: Draw a random number from a uniform distribution: $p_s = U[0, 1]$.
15: **if** $p_s \leq d_i(t)$ **then**
16: Set $\beta_i(t) = 0$.
17: Transfer tasks in $Q_i(t)$ to other MECSs by calling *task_transfer()* in Algorithm 2.
18: **else**
19: Set $\beta_i(t) = 1$.
20: Send $\beta_i(t)$ to neighboring MECS j, $\forall j \in N_i$.

After acquiring an updated Delta activation level $d_i(t)$ (line 13), a MECS i stochastically determines whether to sleep during time slot t (Lines 14–20). After making a sleep decision, MECS i sends $\beta_i(t)$ to the neighboring MECSs.

When MECS i decides to sleep, it transfers the tasks in $Q_i(t)$ to the neighboring MECSs. The task transfer process is presented in Algorithm 2. As reported in [32], unnecessary task transfers ensue after a load of a server is transferred to the least-loaded neighbor because this neighbor may become overloaded by accommodating too much load from its neighbors. To avoid such an adverse situation, the tasks in MECS i must be distributed to each neighboring MECS that decides to be active, and the number of tasks transferred from MECS i to MECS j should be determined to be inversely proportional to the utilization of MECS j. Among the neighboring MECSs of MECS i, we denote the set of MECSs that are active during time slot t as $N_i^{ON}(t)$. MECS i determines the proportion of tasks that will be transferred to MECS $j \in N_i^{ON}(t)$ as follows:

$$\eta_{i,j}(t) = \frac{1/\mu_j(t)}{\sum_{k \in N_i^{ON}(t)} 1/\mu_k(t)}. \tag{9}$$

Subsequently, MECS i transfers $\eta_{i,j}(t) Q_i(t)$ number of tasks to MECS j. When MECS j receives tasks from MECS i, the length of the queue increases from $Q_j(t)$ to $Q_j(t) + \eta_{i,j}(t) Q_i(t)$. If $Q_j(t) + \eta_{i,j}(t) Q_i(t) \geq q_{th}$, the optimization constraints are violated. Therefore, in this case, MECS j informs MECS i of the situation by sending a negative notification to i to force MECS i to remain active during time slot t.

Algorithm 2 Task transfer algorithm

1: **At the beginning of a time slot *t*:**
2: MECS i collects $\beta_j(t)$s from all $j \in N_i$.
3: Determine $N_i^{ON}(t) = \{j : j \in N_i \wedge \beta_j(t) = 1\}$.
4: **while** $j \in N_i^{ON}(t)$ **do**
5: $\eta_{i,j}(t) = \frac{1/\mu_j(t)}{\sum_{k \in N_i^{ON}(t)} 1/\mu_k(t)}$.
6: Send $\eta_{i,j}(t)Q_i(t)$ amount of tasks to a MECS j.
7: **if** Receive a negative notification from j **then**
8: Remain active ($\beta_i(t) = 1$).

Our sleep decision algorithm converges the sleep-control vector to a stable point. This is attributed to the fact that this algorithm can regulate the number of sleeping MECSs even though it operates in a distributed manner. More specifically, MECS j (deciding to be active) prevents MECS i, which decides to sleep from going to sleep when the length of the queue of MECS j becomes larger than q_{th} (if it accommodates the tasks from MECS i).

4.2. MECS Activation Process

The workload imposed on active MECSs increases with the increase in the number of sleeping MECSs, which increases the task queue of the active MECS. Therefore, if an active MECS is likely to violate the constraint associated with the task service delay, the MECS must wake up the neighboring MECSs that are in the sleep state. The task arrival process is stochastic, the dynamics of which are unknown in advance. Therefore, to reduce unnecessary wake-up, we assign a threshold value q_{th}^{on} to an active MECS to determine whether to wake up sleeping MECSs. Let us denote the maximum task queue size of a MECS i as Q_i^{max}. Then, if $Q_i(t)/Q_i^{max} > q_{th}^{on}$ at the start of time slot t, an active MECS i considers that it is highly likely to violate the task service latency during a time slot. Thus, an active MECS starts the wake-up process when the following condition is satisfied at the beginning of a time slot.

$$\frac{Q_i(t)}{Q_i^{max}} > q_{th}^{on}. \tag{10}$$

The threshold value q_{th}^{on} is an operational parameter that can be configured by a MEC system operator. This value results in a trade-off between the amount of energy saved and the possibility of quality of service (QoS) violation. As q_{th}^{on} increases, the sleeping time of a MECS increases because the time at which an active MECS wakes up a sleeping MECS is suspended. Therefore, the amount of energy saved increases with the increase in q_{th}^{on} while the possibility of QoS violation increases as q_{th}^{on} increases.

In this study, we adopt a conservative approach to the MECS activation process. We assume that a MEC system operator satisfying the QoS constraint (i.e., the length of the task queue) is more important than the amount of saved energy. Therefore, when an active MECS decides to begin the activation process, it wakes up all the sleeping neighbors and distributes the tasks in its task queue fairly to all the MECSs that were woken up to reduce the risk of QoS violation. Specifically, if a MECS i wakes up $n_i^w = |N_i - N_i^{ON}(t)|$ sleeping neighbors, the MEC transfers $Q_i(t)/(n_i^w + 1)$ number of tasks to the MECSs that were woken-up.

4.3. Algorithm Complexity

The time complexity of our bio-inspired sleep decision algorithm presented in Algorithm 1 is analyzed as follows. At the beginning of each time slot, a MECS i collects μ_j and d_j for all $j \in N_i$ and calculates their averages (line 3 and line 4). Thus, the time complexity of the averaging operation becomes $O(|N_i|)$. The lines from 7 to 11 determine the Delta activation level. Since it takes at most K iterations before the Delta activation level is decided, its time complexity is less than or equal to $O(K)$. In [33], it is shown that the

Delta–Notch signaling model converges to a stable state after a few rounds. To investigate the temporal behavior, we measure $\delta_d = d'_i(k+1) - d'_i(k)$ (line 10) for different MECSs at different time slots under the default simulation environment presented in Section 5. We depict the result in Figure 2. We can observe that the Delta activation level converges (i.e., $\delta_d \approx 0$) exponentially fast (i.e., after 3 iterations). This behavior is consistent with that in [21] in that the Delta–Notch model stabilizes exponentially.

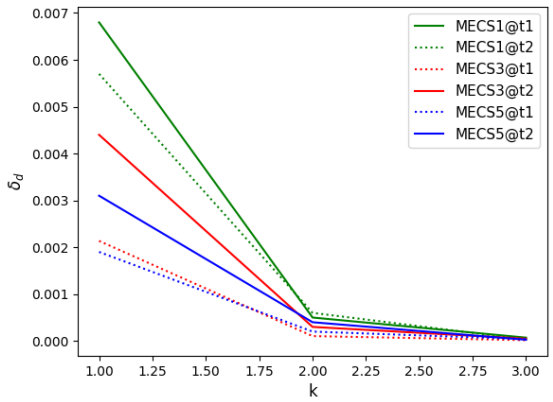


Figure 2. Convergence property of the Delta activation level (the time slots $t1$ and $t2$ are randomly selected and $t1 \neq t2$).

We also note that K is a constant and not related to a system size such as the number of MECSs and the number of tasks. Since K can be configured as a small positive integer, we can say that the worst-case time complexity of the operation determining the Delta activation level (lines 7–11) is $O(K) = O(1)$. The other lines in Algorithm 1 are assignment operations or comparison operations whose time complexity is $O(1)$. Therefore, the worst-case time complexity of our bio-inspired sleep decision algorithm presented in Algorithm 1 is $O(|N_i|)$.

In our task transfer algorithm presented in Algorithm 2, N_i^{ON} is determined by examining β_js sent from all the neighboring MECSs ($j \in N_i$). Therefore, its time complexity becomes $O(|N_i|)$. In lines 4 to 8 in Algorithm 2, tasks are transferred from a MECS deciding to sleep to its active neighbors. Therefore, its time complexity is $O(|N_i^{ON}|)$. Since $N_i^{ON} \subseteq N_i$, the worst-case time complexity of our task transfer algorithm becomes $O(|N_i|)$.

In our MECS activation process, an active MECS compares its queue length with a given threshold to determine whether to wake up its sleeping neighbors or not. This comparison operation takes $O(1)$. Once a MECS i decides to wake up its sleeping neighbors, it wakes up all of the sleeping neighbors and fairly distributes the tasks in its task queue to each sleeping neighbor. If we denote the set of sleeping neighbors of an active MECS i as N_i^s, the task distribution operation takes $O(|N_i^s|)$. Since $N_i^s \subseteq N_i$, the worst-case time complexity of our MECS activation algorithm becomes $O(|N_i|)$. Therefore, the worst-case time complexity of our sleep control mechanisms is $O(|N_i|)$.

5. Performance Evaluation

In this section, we analyze the performance of the proposed method through simulation studies. The default parameter values are as follows, if not specified otherwise. We deploy $|N| = 10$ MECSs randomly. We set $|N_i| = 6$ and select randomly the member of each N_i from N. We set $Q_{max} = 100$, $q_{th}^{ON} = 0.7$, $P_m = 20$ Watt, and $\alpha = 0.5$. We also configure $b_{i,j} = b_i = 300$ Mbps. We vary the workload (ρ) imposed on a MEC system. The workload is defined as a proportion of the total system capacity $\sum_{i \in N} b_i$ during each time slot. Given ρ, $a_{1,i}(t)$ is configured randomly as $\frac{1}{|N|} \sum_{i \in N} b_i \rho U[0,1]$, where $U[0,1]$ is the Uniform distribution.

5.1. Performance Comparison

We compared the performance of the proposed method with those of [7] which use Lyapunov optimization framework. Henceforth, we name the method in [7] as REJO.

In Figure 3, we compare the total energy consumption of a MEC system for different workloads. As the workload imposed on a system increases, the total energy consumed by MECSs increases. However, we observe in the figure that our method can reduce the total energy consumption for all the workloads. For example, when $\rho = 1.0$ and $|N| = 10$, our method reduces the total energy consumption by more than 13%.

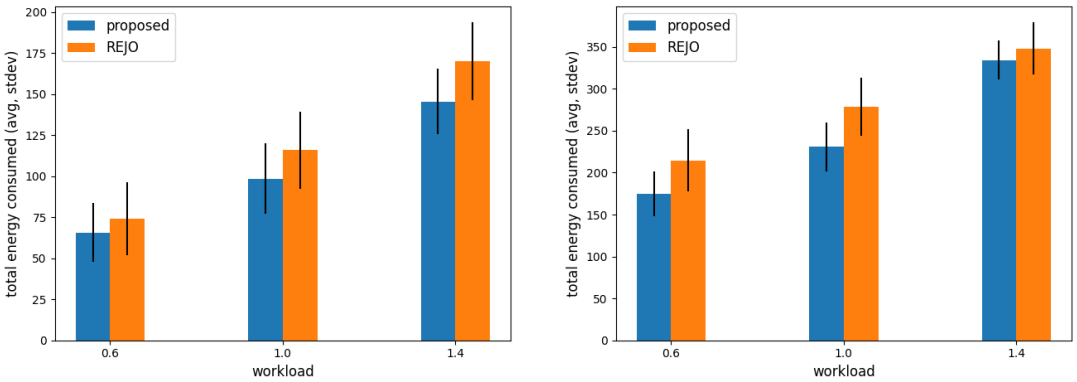


Figure 3. Comparison of energy consumption in a MEC system. (**a**) 10 MECSs; (**b**) 20 MECSs.

In Figure 4, we compare the average service delay of our method with that of REJO. The task service delay obtained by our method is longer than that when REJO is used. However, when we examine the y-axis, the difference is marginal. We also show the standard deviation of the service delay in Figure 4. We observe that our method reduces the standard deviation compared with REJO. For example, when $\rho = 1.0$ and $|N| = 10$, our method reduces the variations in the service delay by more than 35%.

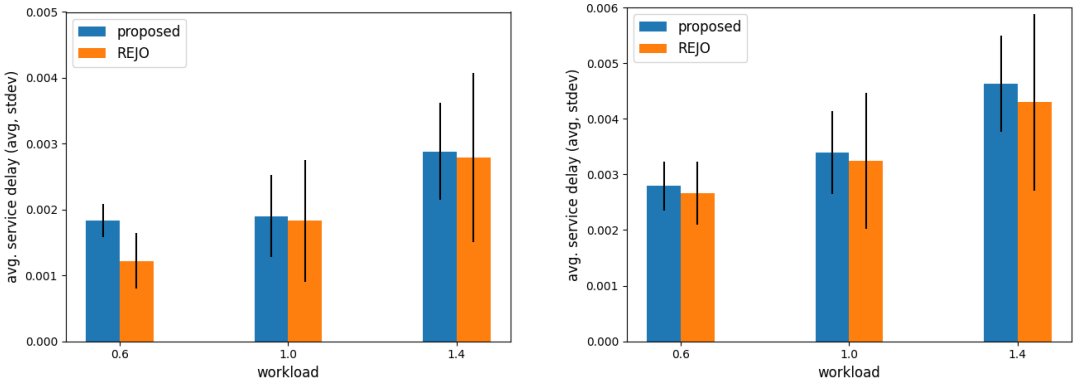
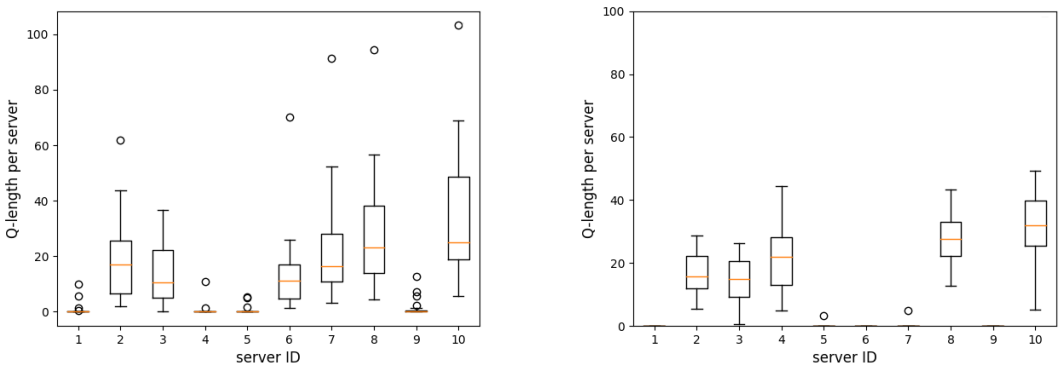


Figure 4. Comparison of service delays. (**a**) 10 MECSs; (**b**) 20 MECSs.

To inspect the reason, we investigate the distribution of each Q_i ($i \in N$) when $\rho = 1.0$ and $|N| = 10$. We show the result in Figure 5. To make the comparison easier, we purposely configure the range of the y-axis in Figure 5a to be the same as that in Figure 5b.

Since our method makes a sleep decision by using the relative load level, our method reduces the first to the third quartile values of each MECS's Q_i. Our method also reduces the number of outliers in each Q_i. By reducing the variation in Q_i, a MEC system can provide a more stable service delay when our method is used than when REJO is used.

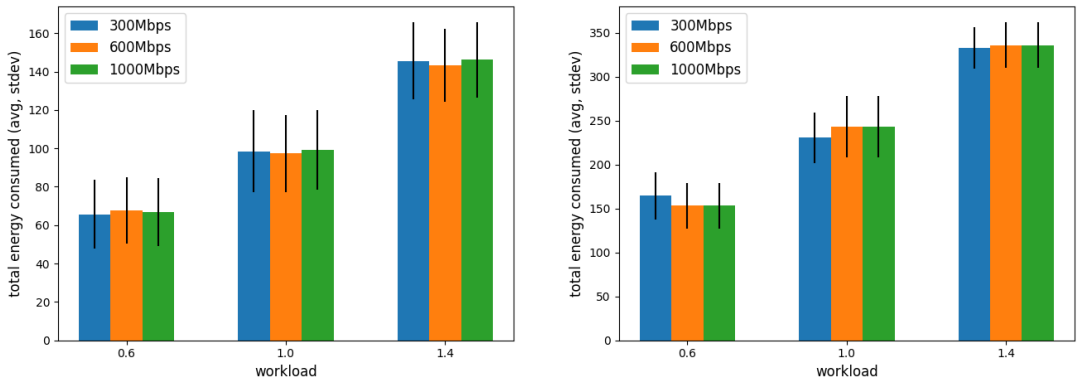


(**a**) (**b**)

Figure 5. Distribution of the tasks queue length ($|N| = 10$, $\rho = 1.0$). (**a**) REJO; (**b**) Proposed.

5.2. Impact of Backhaul Network

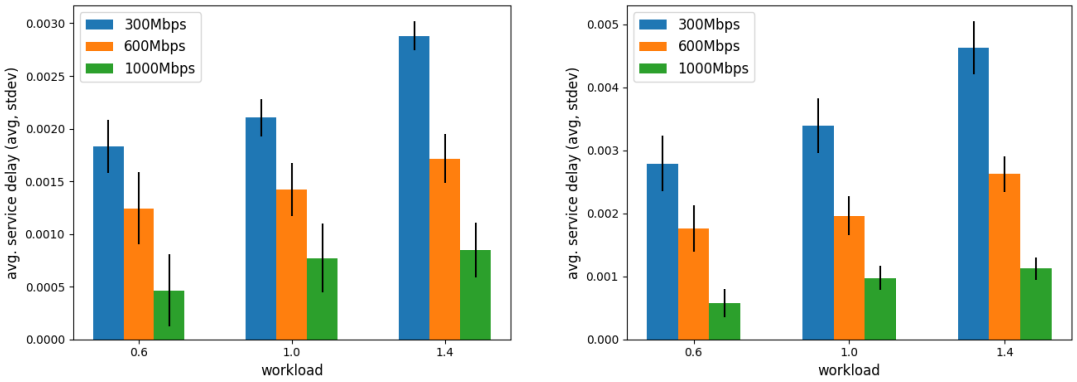
In this section, we inspect the influence of the backhaul network bandwidth (i.e., $b_{i,j}$) on the performance of the sleep control mechanism. We set $b_i = 300$ Mbps and vary $b_{i,j}$. In Figure 6, we show the total energy consumed in a MEC system for various $b_{i,j}$s (300, 600, and 1000 Mbps). We can observe in the figure that the effect of $b_{i,j}$ on the total energy consumed in a MEC system is not so significant. The energy consumed by a MECS is subdivided into the energy consumed for processing tasks (e_i^p) and the energy to transfer tasks to other MECSs (e_i^{tr}). e_i^{tr} is influenced by the number of tasks transferred from a MECS to its neighbors, which does not depend on $b_{i,j}$. In the case of e_i^p, it increases with the utilization of a MECS and μ_i is affected by $a_i = a_{1,i} + a_{2,i}$. The tasks transferred among MECSs affect $a_{2,i}$. However, even though tasks are distributed among the MECSs in an uneven manner, the total number of tasks imposed on a system does not change. Therefore, the total energy consumed in a MEC system is not affected by $b_{i,j}$.



(**a**) (**b**)

Figure 6. Impact of the backhaul network bandwidth on the total energy consumption. (**a**) 10 MECSs; (**b**) 20 MECSs.

Figure 7 shows the impact of $b_{i,j}$ on the average task service delay. As can be seen in the figure, the average task service delay decreases as the bandwidth of a backhaul network increases. The tasks service delay incurred by a MECS i during a time slot t is $d_i^p(t) = (Q_i(t) + a_i(t))/b_i$. When a MECS i decides to sleep, it transfers the tasks in its task queue to its active neighbors, which causes a task transfer delay $d_i^{tr} = \eta_{i,j} Q_i / b_{i,j}$. As $b_{i,j}$ increases, d_i^{tr} decreases. Therefore, the average task service delay in a MEC system decreases as the communication bandwidth of a backhaul network increases.

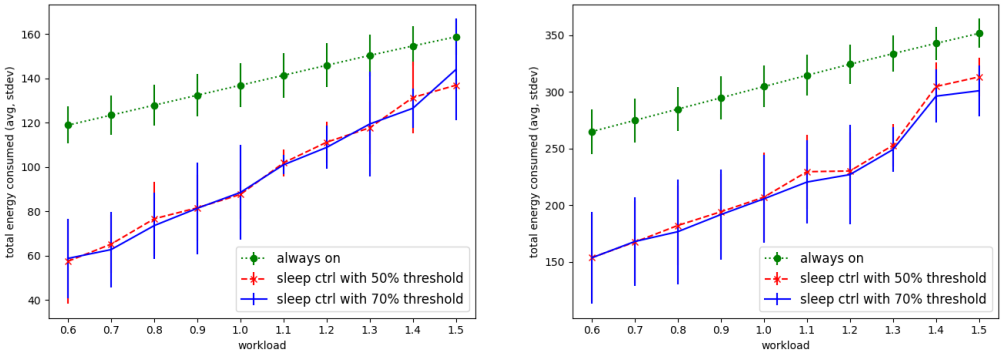


(**a**) (**b**)

Figure 7. Impact of network bandwidth on the task service delay. (**a**) 10 MECSs; (**b**) 20 MECSs.

5.3. Parameter Effect

In this subsection, we present the performance of our sleep control mechanisms in various N,$|N_i|$, ρ, and q_{th}^{ON} when the bandwidth of a backhaul network is much higher than the computing capability of a MECS (i.e., $b_{i,j} >> b_i$). In Figure 8, we display the total energy consumption in a scenario, in which all MECSs can directly communicate with each other. The workload rate of the system varied from 0.6 to 1.5. We used ten and twenty MECSs in the system. The results revealed that the proposed method reduced the total energy consumption by approximately 31%, on average, compared with the case where no sleep control is used. However, the variance in the energy consumed was larger than that when sleep control was not used. This is because the difference in energy consumption between the active and the sleeping MECSs increased when sleep control was activated. In the figure, *sleep ctrl with a 50% threshold* and *sleep ctrl with a 70% threshold* indicate that q_{th}^{ON} is set to 50% and 70% of Q_{max}, respectively. In our experiments, we set Q_{max} to 100. We observed that the energy consumed in the network with different values of q_{th}^{ON} were similar.



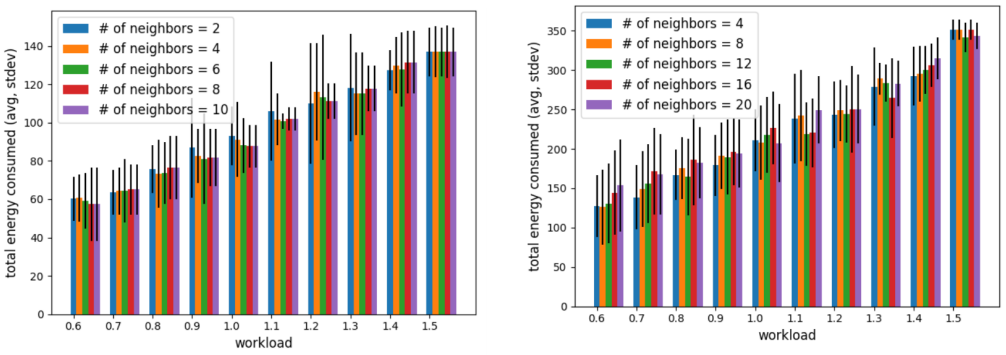
(**a**) (**b**)

Figure 8. Total energy consumed when all MECSs were able to directly communicate with each other. (**a**) when the number of MECSs is 10; (**b**) when the number of MECSs is 20.

In Figures 9 and 10, we display the total energy consumed in the system when $|N_i| \leq |N|$. In these figures, we configure q_{th}^{ON} as 0.5 or 0.7. When the range of the neighborhoods (i.e., $|N_i|$) changes, the average and the standard deviation of the energy consumption do not differ significantly. A similar pattern is observed when the number of MECSs increases. The standard deviation of the energy consumption becomes larger as q_{th}^{ON} increases. This is attributed to that q_{th}^{ON} determines the time when an active MECS wakes up sleeping neighbors.

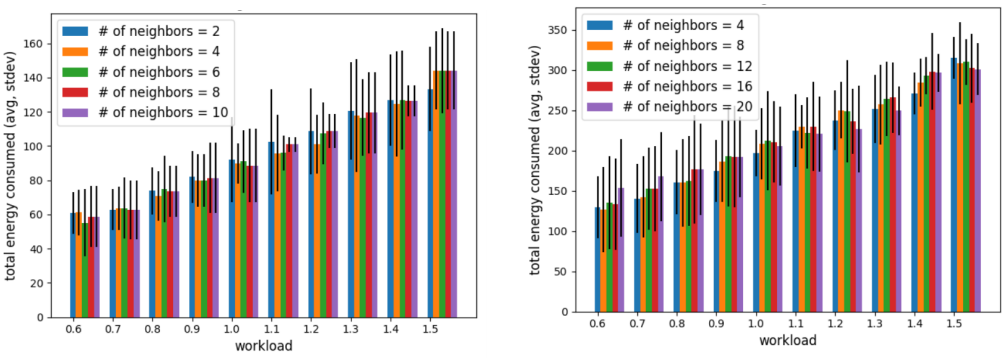
In Figures 11 and 12, we compare the Q_is under different workloads for a neighborhood range of four and six. In the figures, q_{th}^{ON} is set to 0.5 or 0.7 for the MECS under

sleep control. With each q_{th}^{ON}, we measure the length of the queue of each MECS when the workload is 1.0 or 1.5. The first to the third quartile values of each Q_i are located below 50% or 70% of Q_{max}. As the workload increases, the sleeping MECS numbers decrease, and the non-sleeping MECS loads become relatively even. When q_{th}^{ON} is 0.7, the sleeping MECS numbers decrease with respect to the case in which $q_{th}^{ON} = 0.5$, and the standard deviation of the queue length decreases.



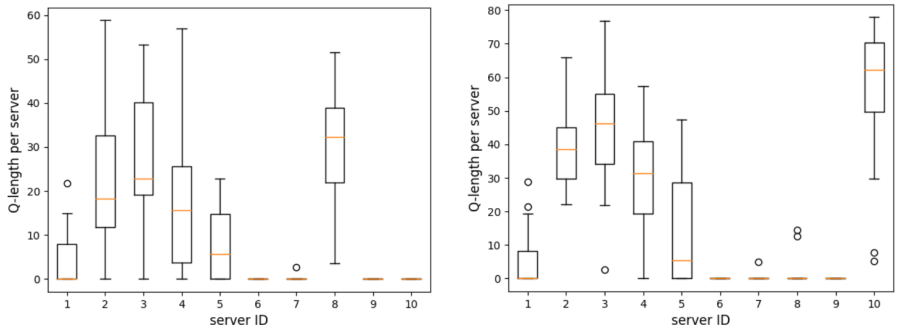
(**a**) (**b**)

Figure 9. Total energy consumption when the neighborhood range varies and $q_{th} = 0.5$. (**a**) The number of MECSs is 10; (**b**) The number of MECSs is 20.



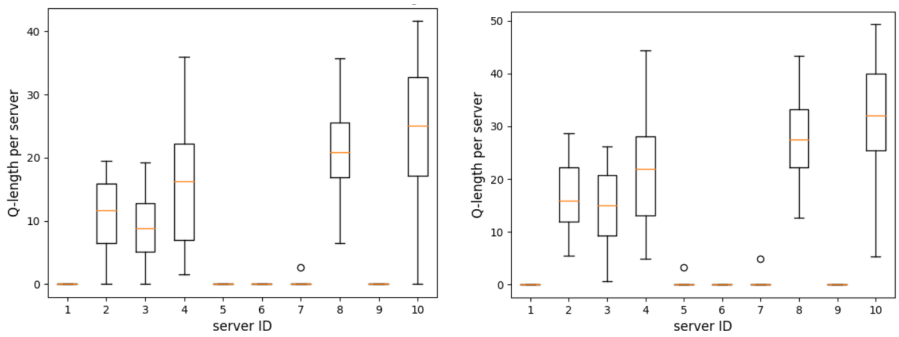
(**a**) (**b**)

Figure 10. Total energy consumption when the neighborhood range varies and $q_{th} = 0.7$. (**a**) The number of MECSs is 10; (**b**) The number of MECSs is 20.



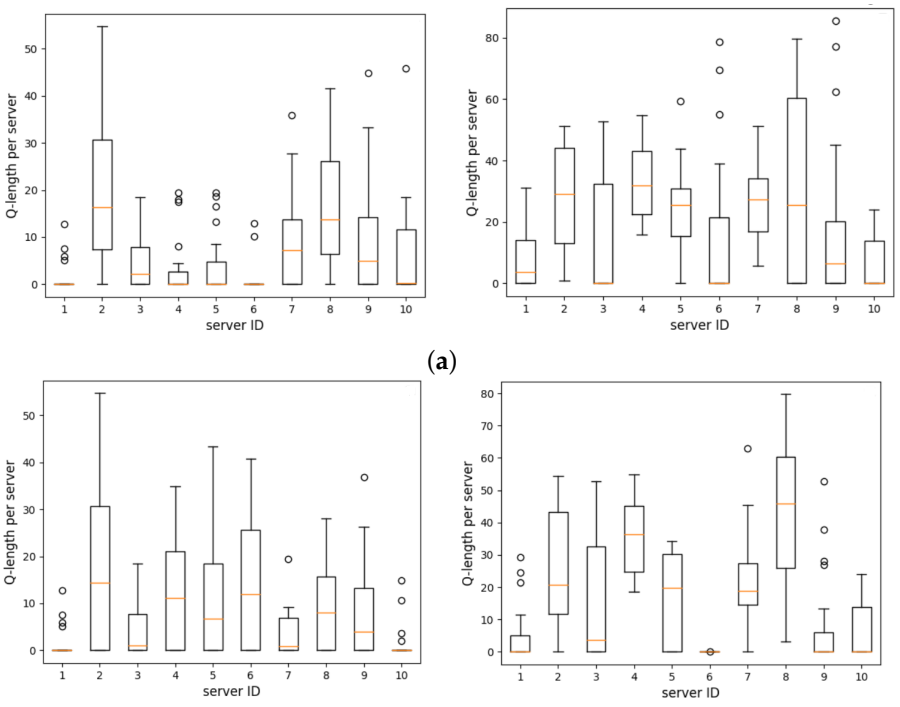
(**a**)

Figure 11. *Cont.*



(**b**)

Figure 11. Length of the queue per server for 10 MECSs with workload rate = 1.0. (**a**) The neighborhood range is 4 (q_{th} = 0.5 or 0.7); (**b**) The neighborhood range is 6 (q_{th} = 0.5 or 0.7).



(**a**)

(**b**)

Figure 12. Length of the queue per server for 10 MECSs with workload rate = 1.5. (**a**) The neighborhood range is 4 (q_{th} = 0.5 or 0.7); (**b**) The neighborhood range is 6 (q_{th} = 0.5 or 0.7).

6. Conclusions and Future Work

In this study, we addressed the sleep control problem pertaining to MECS in a multi-access edge computing system to reduce energy consumption while maintaining task service delay at a reasonable level. To reduce the complexity of the optimization-based algorithms, we adopted the Delta–Notch inter-cell signaling model as a lightweight distributed approach. We evaluated the performance of the proposed approach and analyzed the impact of parameters, such as the range of the neighborhood, workload rate, and threshold to determine the activation of sleep control. The experimental results revealed that compared with a conventional method, the proposed method reduces energy consumption in a MEC system by more than 13% while providing a comparable service delay. In addition, our method reduces the variations in the service delay by more than 35%.

As our future works, we will investigate the optimal wake-up policy that determines the set of MECSs to be woken up and the number of tasks to be transferred to the woken-up MECSs. In addition, we will scrutinize the impact of heterogeneous types of tasks with different QoS requirements on our sleep control mechanism.

Author Contributions: Conceptualization, J.P.; Software, Y.L.; Writing—original draft, J.P.; Writing—review & editing, Y.L. All authors have read and agreed to the published version of the manuscript.

Funding: This study was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (no. 2022R1F1A1065371). The present research was conducted using a research grant from Kwangwoon University in 2022.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. 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]
2. Rodrigues, T.K.; Suto, K.; Nishiyama, H.; Liu, J.; Kato, N. Machine Learning Meets Computation and Communication Control in Evolving Edge and Cloud: Challenges and Future Perspective. *IEEE Commun. Surv. Tutor.* **2020**, *22*, 38–65. [CrossRef]
3. Filali, A.; Abouaomar, A.; Cherkaoui, S.; Kobbane, A.; Guizani, M. Multi-Access Edge Computing: A Survey. *IEEE Access* **2020**, *8*, 197017–197046. [CrossRef]
4. Freitag, C.; Lee, M.B.; Widdicks, K.; Knowles, B.; Blair, G.; Friday, A. The Climate Impact of ICT: A Review of Estimates, Trends and Regulations, Lancaster University. *arXiv* **2020**, arXiv:2102.02622.
5. Wang, S.; Zhang, X.; Yan, Z.; Wenbo, W. Cooperative Edge Computing with Sleep Control under Nonuniform Traffic in Mobile Edge Networks. *IEEE Internet Things J.* **2019**, *6*, 4295–4306. [CrossRef]
6. Liang, C.; Hey, Y.; Yu, F.R.; Zhao, N. Energy-Efficient Resource Allocation in Software-Defined Mobile Networks with Mobile Edge Computing and Caching. In Proceedings of the 2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Atlanta, GA, USA, 1–4 May 2017.
7. Chen, L.; Zhou, S.; Xu, J. Energy Efficient Mobile Edge Computing in Dense Cellular Networks. In Proceedings of the 2017 IEEE International Conference on Communications (ICC), Paris, France, 21–25 May 2017.
8. Zheng, C.; Sicker, D.C. A Survey on Biologically Inspired Algorithms for Computer Networking. *IEEE Commun. Surv. Tutor.* **2013**, *15*, 1160–1191. [CrossRef]
9. Hamrioui, S.; Lorenz, P. Bio Inspired Routing Algorithm and Efficient Communications within IoT. *IEEE Netw.* **2017**, *31*, 74–79. [CrossRef]
10. 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. [CrossRef]
11. Wu, J.; Wang, L.; Pei, Q.; Cui, X.; Liu, F.; Yang, T. HiTDL: High-Throughput Deep Learning Inference at the Hybrid Mobile Edge. *IEEE Trans. Parallel Distrib. Syst.* **2022**, *33*, 4499–4514. [CrossRef]
12. Chen, S.; Jiao, L.; Liu, F.; Wang, L. EdgeDR: An Online Mechanism Design for Demand Response in Edge Clouds. *IEEE Trans. Parallel Distrib. Syst.* **2022**, *33*, 343–358. [CrossRef]
13. Gao, B.; Zhou, Z.; Liu, F.; Xu, F.; Li, B. An Online Framework for Joint Network Selection and Service Placement in Mobile Edge Computing. *IEEE Trans. Mob. Comput.* **2022**, *21*, 3836–3851. [CrossRef]
14. Chen, Q.; Zheng, Z.; Hu, C.; Wang, D.; Liu, F. On-Edge Multi-Task Transfer Learning: Model and Practice with Data-Driven Task Allocation. *IEEE Trans. Parallel Distrib. Syst.* **2020**, *31*, 1357–1371. [CrossRef]
15. Pan, L.; Wang, L.; Chen, S.; Liu, F. Retention-Aware Container Caching for Serverless Edge Computing. In Proceedings of the IEEE Conference on Computer Communications (INFOCOM'22), London, UK, 2–5 May 2022.
16. Chen, S.; Wang, L.; Liu, F. Optimal Admission Control Mechanism Design for Time-Sensitive Services in Edge Computing. In Proceedings of the IEEE Conference on Computer Communications (INFOCOM'22), London, UK, 2–5 May 2022.
17. Pei, Q.; Chen, S.; Zhang, Q.; Zhu, X.; Liu, F.; Jia, Z.; Wang, Y.; Yuan, Y. CoolEdge: Hotspot-relievable Warm Water Cooling for Energy-efficient Edge Datacenters. In Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'22), Lausanne, Switzerland, 28 February–4 March 2022.
18. Jin, P.; Fei, X.; Zhang, Q.; Liu, F.; Li, B. Latency-aware VNF Chain Deployment with Efficient Resource Reuse at Network Edge. In Proceedings of the IEEE Conference on Computer Communications (INFOCOM'20), Toronto, ON, Canada, 6–9 July 2020.
19. Li, M.; Zhang, Q.; Liu, F. Finedge: A Dynamic Cost-efficient Edge Resource Management Platform for NFV Network, In Proceedings of the IEEE/ACM 28th International Symposium on Quality of Service (IWQoS'20), Hang Zhou, China, 15–17 June 2020.
20. Yang, Y.; Hu, Y.; Gursoy, M.C. Deep Reinforcement Learning and Optimization Based Green Mobile Edge Computing. In Proceedings of the 2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, USA, 9–12 January 2021.

21. Zhu, N.; Xu, X.; Han, S.; Lv, S. Sleep-Scheduling and Joint Computation-Communication Resource Allocation in MEC Networks for 5G IIoT. In Proceedings of the 2021 IEEE Wireless Communications and Networking Conference (WCNC), Nanjing, China, 29 March–1 April 2021.
22. Bolla, R.; Bruschi, R.; Carrega, A.; Davoli, F.; Lombardo, C. Trading off Power Consumption and Delay in the Execution of Network Functions by Dynamic Activation of Processing Units. In Proceedings of the 2022 IEEE 8th International Conference on Network Softwarization (NetSoft), Milan, Italy, 27 June–1 July 2022.
23. Wu, B.; Zeng, J.; Shao, S.; Ni, W.; Tang, Y. New Game-Theoretic Approach to Decentralized Path Selection and Sleep Scheduling for Mobile Edge Computing. *IEEE Trans. Wirel. Commun.* **2022**, *21*, 6125–6140. [CrossRef]
24. Ranadheera, S.; Maghsudi, S. Ekram Hossain, Computation Offloading and Activation of Mobile Edge Computing Servers: A Minority Game. *IEEE Wirel. Commun. Lett.* **2018**, *7*, 688–691. [CrossRef]
25. Fragkos, G.; Lebien, S.; Tsiropoulou, E.E. Artificial Intelligent Multi-Access Edge Computing Servers Management. *IEEE Access* **2020**, *8*, 171292–171304. [CrossRef]
26. Badri, H.; Bahreini, T.; Grosu, D.; Yang, K.s. Energy-Aware Application Placement in Mobile Edge Computing: A Stochastic Optimization Approach. *IEEE Trans. Parallel Distrib. Syst.* **2020**, *31*, 909–922. [CrossRef]
27. Zhang, Z.; Long, K.; Wang, J.; Dressler, F. On Swarm Intelligence Inspired Self-Organized Networking: Its Bionic Mechanisms, Designing Principles and Optimization Approaches. *IEEE Commun. Surv. Tutor.* **2014**, *16*, 513–537. [CrossRef]
28. Bitam, S.; Mellouk, A.; Zeadally, S. Bio-Inspired Routing Algorithms Survey for Vehicular Ad Hoc Networks. *IEEE Commun. Surv. Tutor.* **2015**, *17*, 843–867. [CrossRef]
29. Saleem, K.; Alabduljabbar, G.M.; Alrowais, N.; Al-Muhtadi, J.; Imran, M.; Rodrigues, J.J.P.C. Bio-Inspired Network Security for 5G-Enabled IoT Applications. *IEEE Access* **2020**, *8*, 229152–229160. [CrossRef]
30. Collier, J.R.; Monk, N.A.; Maini, P.K.; Lewis, J.H. Pattern Formation by Lateral Inhibition with Feedback: A Mathematical Model of Delta–Notch Inter-Cellular Signalling. *J. Theor. Biol.* **1996**, *183*, 429–446. [CrossRef]
31. Koon, Y.L.; Zhang, S.; Rahmat, M.B.; Koh, C.G.; Chiam, K. Enhanced Delta–Notch Lateral Inhibition Model Incorporating Intracellular Notch Heterogeneity and Tension-Dependent Rate of Delta–Notch Binding that Reproduces Sprouting Angiogenesis Patterns. *Sci. Rep.* **2018**, *8*, 9519. [CrossRef] [PubMed]
32. Park, J.; Kim, Y.; Lee, J. Mobility Load Balancing Method for Self-Organizing Wireless Networks Inspired by Synchronization and Matching With Preferences. *IEEE Trans. Veh. Technol.* **2018**, *67*, 2594–2606. [CrossRef]
33. Liu, F.; Sun, D.; Murakami, R.; Matsuno, H. Modeling and Analysis of the Delta–Notch Dependent Boundary Formation in the Drosophila Large Intestine. *BMC Syst. Biol.* **2017**, *11*, 43–60. [CrossRef] [PubMed]