Lyapunov-Guided Delay-Aware Energy Efficient Offloading in IIoT-MEC Systems

Huaming Wu, *Senior Member, IEEE*, Junqi Chen, Tu N. Nguyen, *Senior Member, IEEE*, and Huijun Tang

***Abstract*—With the increasingly humanized and intelligent operation of Industrial Internet of Things (IIoT) systems in Industry 5.0, delay-sensitive and compute-intensive (DSCI) devices have proliferated, and their demand for low latency and low power consumption has become more and more eager. In order to extend the battery life and improve the quality of user experience, we can offload DSCI-type workloads to mobile edge computing (MEC) servers for processing. However, offloading massive amounts of tasks will incur higher energy consumption, which is a severe test for the limited battery capacity of devices. In addition, the delay caused by frequent communication between IIoT devices and MEC cannot be ignored. In this article, we first formulate the stochastic computation offloading problem to minimize long-term energy consumption. Then, we construct a virtual queue using perturbed Lyapunov optimization techniques to transform the problem of guaranteeing task deadlines into a stable control problem for the virtual queue. Based on this, a novel delay-aware energy-efficient (DAEE) online offloading algorithm is proposed, which can adaptively offload more tasks when the network quality is good. Meanwhile, it delays transmission in the case of poor connectivity but ensures that the deadline is not violated. Moreover, we theoretically demonstrated that DAEE can enable the system to achieve an energy-delay tradeoff, and analyzed the feasibility of constructing virtual queues to assist the actual queue offloading tasks. Finally, simulation results show that DAEE performs well in minimizing energy consumption and maintaining low latency, especially for DSCI-type tasks.**

***Index Terms*—Delay-sensitive and compute-intensive (DSCI)-type tasks, energy efficient, Industrial Internet of Things (IIoT), mobile edge computing (MEC), workload offloading.**

Manuscript received 11 June 2022; revised 24 July 2022 and 20 August 2022; accepted 7 September 2022. Date of publication 15 September 2022; date of current version 13 December 2022. This work was supported by the National Natural Science Foundation of China under Grant 62071327. Paper no. TII-22-2495. *(Corresponding author: Huaming Wu.)*

Huaming Wu and Junqi Chen are with the Center for Applied Mathematics, Tianjin University, Tianjin 300072, China (e-mail: whming@tju.edu.cn; junqichen@tju.edu.cn).

Tu N. Nguyen is with the Department of Computer Science, Kennesaw State University, Marietta, GA 30060 USA (e-mail: tu.nguyen@kennesaw.edu).

Huijun Tang is with the School of Cyberspace, Hangzhou Dianzi University, Hangzhou 310018, China (e-mail: tanghuijune@hdu.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TII.2022.3206787.

Digital Object Identifier 10.1109/TII.2022.3206787

I. Introduction

THE Industrial Internet of Things (IIoT) is an important network for modern industrial enterprises to move toward automation and digitization in Industry 5.0, which has brought value-added services to many fields such as smart manufacturing, smart grid, and intelligent digital supply chains [1]. Nowadays, we have realized seamless communication among people, processes, and objects. The growth in global connection is primarily driven by IIoT devices. The proliferation of connected devices poses a challenge to IIoT networks that rely on frequent communication. How to enhance the capability of IIoT networks to achieve higher business requirements (e.g., high security and low energy consumption) has become a hot topic. The emerging fifth-generation (5G) wireless communication brings together various enabling technologies and has become an important driving force for the comprehensive deployment of the IoT [2]. With the increasingly humanized and intelligent operation of IIoT systems, delay-sensitive and compute-intensive (DSCI) innovative services [3], e.g., intelligent manufacturing, fault diagnosis, intelligent logistics, smart supply chain, industrial cognitive internet of vehicles (CIoV) [4], and Big Data analytics, emerge as the times require. Every task link in IIoT is highly dependent on various IIoT devices, especially to handle DSCI-type tasks, which require more and more computing resources.

Unfortunately, many IIoT devices may only have limited computing capacity or no computing capacity due to current mobile hardware [5]. This will seriously affect the performance of the device and the quality of user experience, especially when DSCI-type tasks are executed on the devices. Due to the limitations of inherent resources (e.g., network bandwidth and computational ability), many complex applications cannot be widely run on terminal devices. This problem has become a bottleneck in improving the quality of service (QoS) [6]. Meanwhile, running DSCI-type tasks on the device is more energy consuming, thereby accelerating the shortening of the device's battery life. Limited battery life increases the maintenance cost of IIoT devices, and the cost of replacing batteries is often higher than the cost of IIoT devices themselves. For instance, in an industrial environment with only 10 000 sensors, the battery needs to be replaced nearly 3 333 times each year [7]. Mobile edge computing (MEC), where data processing can be implemented at edge nodes, can better support the real-time intelligent execution of IIoT services [8]. MEC has become an important computing paradigm in IIoT due to its high efficiency,

high security, and low delay. Therefore, we can transfer the workload to the edge for processing, helping the devices share the heavy work [9] and prolong battery life.

In addition, these DSCI-type applications put forward harsh requirements on the network environment. DSCI-type tasks have more stringent latency requirements. On the one hand, the status of the wireless channel directly affects the transmission energy consumption of IIoT devices, i.e., transferring data in a bad connection will consume more energy [10]. However, greedily choosing a period with better network status to offload workloads can reduce transmission energy consumption, but such a diversion may cause the queue length of the workload in the buffer to be very large, even exceeding the worst-case delay. On the other hand, since the system cannot quickly respond to some burst computing requirements, DSCI-type tasks may experience long queuing delays in a heavy-loaded MEC environment, thus violating their delay requirements [11]. Meanwhile, in the process of real-time wireless network communication, the network quality and channel state are time-varying and random, which can be severely affected by the device location, network congestion, and so on [12], [13]. And the task arrival process of the device is also difficult to obtain. Therefore, in dynamic systems, it is crucial to design an effective task offloading strategy to optimize energy consumption.

To this end, we exploit the Lyapunov optimization to address the challenges of joint resource allocation and task offloading. Lyapunov optimization refers to the use of Lyapunov functions to optimally control dynamic systems [14]. In real-time IoT systems, the state of the wireless channel, the location of the user, and the task generation process are highly dynamic and stochastic. Lyapunov optimization is a powerful tool for task assignment and offloading because it does not require knowledge of the probability distribution of the stochastic event process and has a lower computational complexity [15]. Based on the task offloading strategy optimized by Lyapunov, the usual method is to construct the quadratic function of the task queue backlog as the Lyapunov function and minimize the upper bound of "Lyapunov drift + penalty function" so that the entire system is in a stable state, while optimizing the system performance [16]. Mukherjee et al. [17] designed a hierarchical offload scheduling strategy for tasks with different delay periods. This strategy utilizes the "Lyapunov drift + penalty function" to schedule tasks in the queue to ensure that a greater number of tasks are maximally completed within the task deadline. Guo et al. [18] proposed a Lyapunov-optimized delayed task allocation scheme, which minimizes the system performance by optimizing the task allocation between IoT-edge-cloud. consumption. eTime [19] is an energy-efficient transmission strategy in a multidevice-single-server IoT-Cloud system, suitable for prefetch-friendly and delay-tolerant applications. The algorithm prefetches commonly used data by adaptively selecting the time period when the network connection is good, and delays the transmission of delay-tolerant data when the network quality is poor, achieving an energy-saving effect of 20%–35%. Chen et al. [20] subtly transformed the transmission energy consumption minimization problem into a knapsack problem in a multidevice-single-server IoT-Edge system, and proposed an energy efficient dynamic offloading algorithm (EEDOA) based on Lyapunov optimization. The algorithm adaptively offloads all tasks to the MEC server by optimizing "Lyapunov drift + transmission energy consumption" to achieve the energy-saving goal.

Recently, there are also many studies that combine Lyapunov optimization and artificial intelligence to solve the task offloading problem. Dai et al. [21] used a digital twin network to model network topology and random task arrival in IIoT systems and proposed an asynchronous actor-critic algorithm to minimize the long-term energy efficiency. Zhuang et al. [22] proposed an adaptive network routing method based on deep reinforcement learning (DRL) to solve the problem of bursty data flow. The DRL module can efficiently learn better estimates of long-term Lyapunov drift and penalty functions.

However, most of the existing task allocation and offloading strategies only consider one aspect of compute-intensive or delay-sensitive (tolerant), and few studies consider both compute-intensive and delay-sensitive aspects at the same time. The energy-saving task offloading work based on Lyapunov optimization often cannot effectively perceive the queue backlog status, and it is also insufficient in utilizing real-time network status, so it is not suitable for DSCI-type task offloading. Considering the mobility of the device and the limited computing resources of the MEC are more realistic scenarios.

To cope with the aforementioned challenges, we take the mobility of devices into consideration and design a workload distribution algorithm for DSCI-type applications by using Lyapunov optimization. The purpose of this article is to minimize the long-term energy consumption of the system while reducing the average traffic delay as much as possible. And we propose a novel delay-aware energy-efficient (DAEE) task offloading scheme. The main contributions are summarized as follows.

1) The virtual queue is designed to sense the backlog of the actual task queue and assist the actual queue to utilize the better network to offload tasks to a greater extent. We theoretically proved the feasibility of the cooperation between virtual queues and actual queues.
2) We transform the problem of ensuring task deadlines into a problem of queue stability control. DAEE can adaptively make online offloading decisions and allow the system to progressively tend to optimal energy consumption levels while maintaining a low queue backlog for DSCI-type workloads.
3) To simulate the MEC environment more vividly, we consider the mobility of devices in the model, that is, each device can move freely in a certain area.

II. System Model and Problem Formulation

A. System Overview

The proposed model in an IIoT-Edge system with a base station (BS) is depicted in Fig. 1, which consists of one edge server and N devices, where $\mathcal{N} = \{1, 2, \ldots, N\}$ represents a collection of device indicators. The IIoT scenario here is composed of devices that are highly dependent on the state of the wireless network. Meanwhile, the MEC server is connected

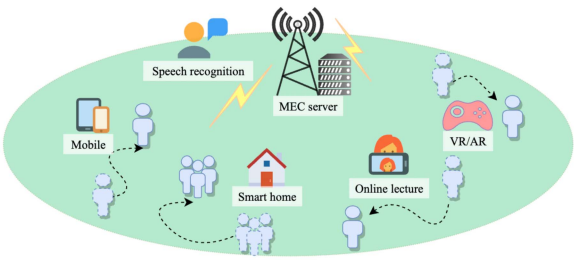

Fig. 1. IIoT-MEC scenario that is highly dependent on the state of the wireless network (users can move freely, and the curve represents the user's movement trajectory).

to these devices via the cellular wireless network, serving N devices.

We assume that the workloads are of DSCI-type and can be arbitrarily divided, which should be transmitted through the wireless network and processed on the MEC server. We establish a time slot system with the slot length τ_0, where $t \in \mathcal{T} = \{0, 1, \ldots, T-1\}$ are discrete. In order to be more consistent with the number of random events occurring per unit time (or space) in the real scene, we use the Poisson distribution with mean $\mathbb{E}\{A_i(t)\} = \lambda_i$ to simulate the generation of IIoT workloads, and the corresponding random variable is $A(t) = (A_1(t), \ldots, A_N(t))$, where $A_i(t)$ is the total amount of workload that the ith IIoT device arrives at the buffer in the time slot t. Normally, $A_i(t)$ follows independent identical distribution (i.i.d.) over time slots. The MEC server and terminal devices are in a certain area where they can be tagged. The devices can move freely within this area, and their movements do not affect each other. We mark the position of device i in time slot t as $(x_i(t), y_i(t))$, and its positions are independent identical distribution in different time slots. That is, $x_i(t) \sim U[0, \mathcal{X}]$, $y_i(t) \sim U[0, \mathcal{Y}]$, $i \in \mathcal{N}$, where $\mathcal{X}$ and $\mathcal{Y}$ are related to the selection of region. In addition, we assume that the location of the MEC server $(x_0(t), y_0(t))$ is fixed. Therefore, the distance between the ith device and the MEC server on the tth time slot is

$$d_i(t) = \| (x_i(t), y_i(t)) - (x_0(t), y_0(t)) \|_2. \tag{1}$$

And it is easy to see that d_i is time varying.

B. Channel Model and Workload Offloading Model

In a real-time wireless network system, the quality of wireless networks and the channel state are random and unpredictable. If we offload a large number of workloads to MEC servers through wireless channels to relieve the pressure on devices themselves, it is very meaningful how to effectively and rationally divert the workload. Meanwhile, we should note that bandwidth resources are often limited. Let $K(t)$ represent the number of available uplink subchannels, and simulate the randomness and time variability of the network by changing the number of available subchannels $K(t)$ in different time slots. Besides, we also assume that wireless channels are i.i.d. and flat block fading, i.e., the channels remain unchanged within a time slot and vary between different time slots [23]. We use time-division multiple access (TDMA) technology to offload tasks in subchannels, that is, the bandwidth of a multiuser time-sharing carrier. In each time slot t, TDMA not only allows multiple devices to access a subchannel at different times, but also allows a single device to access different subchannels, too. To facilitate modeling, we assume that an IIoT device can only access one subchannel at a time [20].

Generally, the computing capacity of the IIoT device itself is insufficient to support the normal operation of DSCI-type tasks. At the same time, local computing may cause more energy consumption than offloading. In our model, the generated workload will be all offloaded to the MEC server for processing. Our idea is to select a time period with better network conditions to offload tasks to help the device. Since it takes more energy to transfer tasks when the connection is poor, greedily choosing a time when the network is in good condition to offload can result in a very large queue length of workload in the device buffer. Therefore, we need to find a suitable diversion method under the limitation of network bandwidth resources.

Let the offloading duration of the ith device on the tth time slot be $l_i(t)$. Here, $l_i(t)$ refers to the amount of time selected for the offloading workload in a time slot, which satisfies as

$$0 \le l_i(t) \le \tau_0 \tag{2}$$

that is to say, in a time slot t, the offloading duration $l_i(t)$ cannot exceed the unit time slot length τ_0. Moreover, we make two assumptions about the model.

1) Assuming that the size of the calculated output result is very small, so we can ignore the delay of feedback [24].
2) Assuming that the MEC server has sufficient computing resources, so the execution delay of the server can be ignored in the model [25].

We further denote $l(t) = \{l_1(t), l_2(t), \ldots, l_N(t)\}$ as an offloading action during the slot t. Depending on the way that terminal devices access the wireless channels, the total offloading duration must not exceed the available bandwidth, shown as follows:

$$\sum_{i=1}^{N} l_i(t) \le K(t) \cdot \tau_0. \tag{3}$$

According to the Shannon–Hartley formula, the offloading rate of the ith device on the tth time slot can be written as

$$r_i(t) = W \log_2 \left(1 + \frac{P_i h_i(t)}{\sigma}\right) \tag{4}$$

where P_i is the transmission power of the ith device, and $h_i(t)$ is the corresponding channel gain. The subchannels have the same bandwidth, denoted as W. σ is the noise power at the MEC server. According to the communication theory [26], the channel gain $h_i(t) = \gamma_i(t) g_0 (\frac{d_0}{d_i(t)})^{\theta}$, where $\gamma_i(t)$ is the small-scale fading channel power gain and d_0 is the relative distance between devices and MEC server. $d_i(t)$, as defined in (1), represents the mobility of different devices. In addition, g_0 is the path-loss constant, and θ is the pass-loss exponent.

Note that $A_i(t)$, $K(t)$, and $r_i(t)$ are all changing with time. Although these variables are almost not predictable, certain

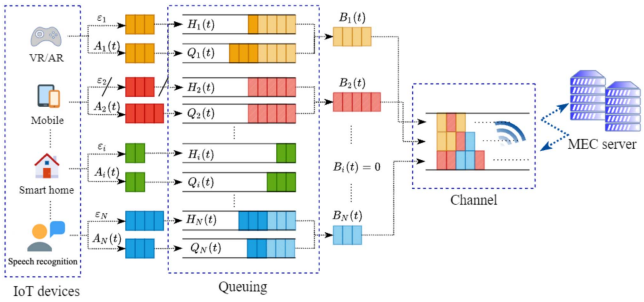

Fig. 2. Architecture of the proposed queueing-based offloading model. Notice that the workloads in the light-colored areas indicate what will be offloaded. Because the queue status of $Q_2(t)$ at this time is empty, there is no task arrival process in the corresponding virtual queue.

methods can be applied to measure specific values in the current time slot.

C. Queuing Model

1) Buffering Actual Queuing Model: The queue-based offloading model proposed in our article is shown in Fig. 2. We divide a workload buffer for each device, and each buffer stores two queues.

The workloads generated by IIoT devices enter the corresponding buffer in a first-in-first-out queuing manner, forming the actual queue $Q(t) = (Q_1(t), \ldots, Q_N(t))$, where $Q_i(t)$ is the existing backlog of tasks for the ith device in the tth time slot. By default, all the queues are empty at the initial moment, i.e., $Q_i(0) = 0$. Define $B_i(t)$ as the amount of offloaded workload in bits, which can be obtained naturally by

$$B_i(t) = r_i(t) \cdot l_i(t). \tag{5}$$

The queue backlog is updated by the following formula:

$$Q_i(t+1) = \max\{Q_i(t) - B_i(t), 0\} + A_i(t). \tag{6}$$

2) Delay-Aware Virtual Queuing Model: Corresponding to each actual queue Q, we develop a virtual queue H as shown in Fig. 2, which is called ε-persistent transmission queue and is dynamically updated according to

$$H_i(t+1) = \max\left\{H_i(t) - B_i(t) + \varepsilon_i 1_{\{Q_i(t)>0\}}, 0\right\} \tag{7}$$

where $0 < \varepsilon_i < \mathbb{E}\{A_i(t)\} = \lambda_i$ is a prespecified constant. $1_{\{Q_i(t)>0\}}$ represents a characteristic function, which only takes the value of 1 when $Q_i(t) > 0$. $H_i(t)$ is constructed using Lyapunov optimization technology. It cooperates with the actual queue $Q_i(t)$ to achieve the purpose of assisting the workload offloading.

From the perspective of the workloads updated methods of the virtual queues in (7), when the actual queue backlogs are not empty, we can draw the following conclusions. First, the offloading amount of the two queues $B_i(t)$ is equal. Second, the virtual queue has a stable task arrival rate ε_i, which is different from $A_i(t)$ and will not change over the time slot. By constructing the virtual queue in this way, such virtual queues are said to be delay aware. The delay-aware virtual queuing model can also ensure a bounded worst-case delay for each application [27]. We can further control the worst-case delay by controlling the length of the two queues to meet the task delay. Details are described in Section III-D.

Due to the fact that the offloading duration $l_i(t)$ in each time slot must not exceed τ_0, and $l_i(t)$ should be tightened to the offloading threshold $T_i(t)$

$$l_i(t) \le T_i(t) = \min\left\{\tau_0, \frac{Q_i(t)}{r_i(t)}, \frac{H_i(t)}{r_i(t)}\right\}. \tag{8}$$

In other words, the offloading duration in each time slot must not exceed the offloading threshold $T_i(t)$.

D. Energy Consumption Model

Generally, IIoT devices do not have enough computing capacity on their own to support the normal operation of DSCI-type tasks. In many cases, the local computation may consume more energy than offloading. As mentioned earlier, in our model, the generated workload will be fully offloaded to the MEC server for processing. So, we do not need to calculate the local execution energy consumption.

Assuming that the MEC server (multicore CPU with extremely high speed) has sufficient computing resources, the execution delay of the server is ignored [25]. So, the total energy consumption of the system mainly comes from offloading the workload, that is, the energy consumption of transmission. We represent the system energy consumption caused by offloading workloads on the tth time slot as $E(t)$, which is jointly determined by the current offloading action $l(t)$ and the transmit power of different devices, defined as follows:

$$E(t) = \sum_{i=1}^{N} P_i \cdot l_i(t). \tag{9}$$

We will show that our framework can adjust the relative importance between the average traffic delay and the average energy consumption.

E. Problem Formulation

It is worth noting that blindly choosing to transmit workloads only during better periods of time on the wireless network can reduce energy consumption to a certain extent, but it may cause the queue backlog to become large and exceed the deadline of tasks, resulting in huge delays and poor user experience.

However, the existence of deadlines forces the system to offload workloads to reduce latency, but it also leads to an increase in energy consumption. Therefore, the process of offloading follows a tradeoff between energy and delay. To further understand this tradeoff, while ensuring that workloads are completed within the deadline, all actual and virtual queues should remain stable for a long time, as shown follows:

$$\bar{Q} = \limsup_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{i=1}^{N} \mathbb{E}\left[Q_i(t)\right] < \epsilon \tag{10}$$

$$\bar{H} = \limsup_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{i=1}^{N} \mathbb{E}\left[H_i(t)\right] < \epsilon \tag{11}$$

where $\bar{Q}$ and $\bar{H}$ represent the actual and virtual queue lengths in the sense of time average, respectively. The longer the length, the longer the offloading time required, and the resulting delay will increase.

The establishment of (10) and (11) implies that all the queue backlogs in the buffer can be offloaded to the MEC server within the required time. Our goal is to minimize the long-term energy consumption within the deadline of workload, while also reducing the delay caused by offloading as much as possible, thereby, improving the quality of experience for users. Based on the aforementioned requirements, we organize it into a stochastic optimization problem $\mathcal{P}_1$ as

$$(\boldsymbol{\mathcal{P}_1}) \quad \min: \quad \bar{E} = \limsup_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}[E(t)]$$
$$\text{s.t.}: \quad (2),\ (3),\ (8),\ (10),\ \text{and}\ (11) \tag{12}$$

where $\bar{E}$ represents the time-averaged energy consumption in the long term. Equation (2) ensures that the offloading duration $l_i(t)$ cannot exceed the unit time slot length τ_0. Equation (8) ensures that the offloading duration in each time slot must not exceed the offloading threshold $T_i(t)$. Equation (3) is the bandwidth constraint ensuring that the total offloading duration must not exceed the available bandwidth. Equations (10) and (11) are delay constraints for the time-averaged actual and virtual queue lengths, respectively.

III. Perturbed Lyapunov-Based DAEE Algorithm

A. Problem Analysis With Perturbed Lyapunov Optimization

By using the perturbed Lyapunov optimization, the problem of ensuring the deadline of workloads is transformed into the problem of the virtual queue stability control. In particular, we define the vector $\Theta(t) = [Q(t), H(t)]$ that is composed of all the queues of the system in time slot t to represent the current backlog state. Similar to the Lyapunov function defined in [28], we have

$$L(\Theta(t)) = \frac{1}{2}\sum_{i=1}^{N}\left\{Q_i(t)^2 + H_i(t)^2\right\} \tag{13}$$

where $L(\Theta(t))$ is a scalar, which represents the total congestion of the two queues and is used to reflect the delay of the workloads. The size of the scalar clearly reflects the state of the queues. We can easily find that if there is a large backlog in any queue, $L(\Theta(t))$ may become very large. So, only when the backlogs of all queues are small, can $L(\Theta(t))$ get a smaller value.

Next, we define $\Delta(\Theta(t))$ as the one-step conditional Lyapunov drift, expressed as

$$\Delta(\Theta(t)) = \mathbb{E}\{L(\Theta(t+1)) - L(\Theta(t)) \mid \Theta(t)\}. \tag{14}$$

To minimize the long-term average energy consumption while satisfying the delay requirement of each device, DAEE should make optimal offloading action $l(t)$ in each slot to minimize the drift-plus-energy expression

$$\Delta(\Theta(t)) + V\mathbb{E}\{E(t) \mid \Theta(t)\} \tag{15}$$

where V is a parameter that adjusts the relative importance between energy and delay. In this way, DAEE can flexibly formulate an offloading action between delay and energy consumption based on real-time scenarios.

Intending to solve such random optimization problems more efficiently, first of all, we have to scale (15) and it is necessary to introduce the following lemma.

Lemma 1: Let the real numbers a_1, a_2, and a_3 be nonnegative, $a = \max[a_3 - a_2, 0] + a_1$, then $a^2 \le a_1^2 + a_2^2 + a_3^2 - 2a_3(a_2 - a_1)$.

The detailed proof is deferred to Appendix A. Recall the definition of the Lyapunov function, applying *Lemma 1*, we have

$$[Q_i(t+1)]^2 \le [Q_i(t)]^2 + [B_i(t)]^2 + [A_i(t)]^2 - 2Q_i(t)\left(B_i(t) - A_i(t)\right) \tag{16}$$

$$[H_i(t+1)]^2 \le [H_i(t)]^2 + [B_i(t)]^2 + \varepsilon_i^2 - 2H_i(t)\left(B_i(t) - \varepsilon_i 1_{\{Q_i(t)>0\}}\right). \tag{17}$$

After substituting (16) and (17) into (13), we can obtain

$$L(\Theta(t+1)) \le \frac{1}{2}\sum_{i=1}^{N}\left\{2[B_i(t)]^2 + [A_i(t)]^2 + \varepsilon_i^2 + [Q_i(t)]^2 + [H_i(t)]^2 - 2Q_i(t)(B_i(t) - A_i(t)) - 2H_i(t)\left(B_i(t) - \varepsilon_i 1_{\{Q_i(t)>0\}}\right)\right\}. \tag{18}$$

According to (14), one-step conditional Lyapunov drift $\Delta(\Theta(t))$ for a general offloading action satisfies

$$\Delta(\Theta(t)) \le D - \sum_{i=1}^{N} Q_i(t)\mathbb{E}\left\{B_i(t) - A_i(t) \mid \Theta(t)\right\} - \sum_{i=1}^{N} H_i(t)\mathbb{E}\left\{B_i(t) - \varepsilon_i 1_{\{Q_i(t)>0\}} \mid \Theta(t)\right\} \tag{19}$$

where $D = \frac{1}{2}\sum_{i=1}^{N}\{2[B_i^{\max}]^2 + [A_i^{\max}]^2 + \varepsilon_i^2\}$, and $A_i^{\max} \ge A_i(t)$ represents the maximum amount of workloads that an IIoT device can generate in a time slot. $B_i^{\max} \ge B_i(t)$ indicates the number of workloads offloaded in a time slot, where $\forall i \in \mathcal{N}$.

Now, the scaled item of Lyapunov drift in (19) only contains relevant variables of the current time slot, which meets the conditions for online decision making. Therefore, the upper bound of the drift-plus-energy expression in (15) can be given by Lemma 2.

Lemma 2: Given that the vector $\Theta(t) = [Q(t), H(t)]$ is the workload backlog status of queues, under any feasible offloading action $l(t)$, the drift-plus-energy item expressed in (15) can be controlled by an upper bound

$$\Delta(\Theta(t)) + V\mathbb{E}\{E(t) \mid \Theta(t)\} \le D - \sum_{i=1}^{N} Q_i(t)\mathbb{E}\left\{B_i(t) - A_i(t) \mid \Theta(t)\right\}$$

$$-\sum_{i=1}^{N} H_i(t)\mathbb{E}\left\{B_i(t)-\varepsilon_i 1_{\{Q_i(t)>0\}} \mid \Theta(t)\right\} + V\mathbb{E}\{E(t)\mid\Theta(t)\} \quad (20)$$

where $D=\frac{1}{2}\sum_{i=1}^{N}\{2[B_i^{\max}]^2+[A_i^{\max}]^2+\varepsilon_i^2\}$.

The detailed proof is deferred to Appendix B.

B. Design of the DAEE Algorithm

Our goal was originally to minimize drift-plus-energy in (15). Because (15) contains the implicit maximum function in (6) and (7). Without undermining the optimality, we minimize its upper bound in (20), which is equivalent to indirectly minimizing itself.

Then, we devise an optimal offloading decision algorithm called DAEE. In addition to energy saving, it can satisfy the backlog of each queue at a lower level. Next, we have the optimization problem $\mathcal{P}_2$ as

$$\begin{aligned}(\boldsymbol{\mathcal{P}_2})\quad \min:\quad & D-\sum_{i=1}^{N} Q_i(t)\mathbb{E}\left\{B_i(t)-A_i(t)\Theta(t)\right\}\\ & -\sum_{i=1}^{N} H_i(t)\mathbb{E}\left\{B_i(t)-\varepsilon_i 1_{\{Q_i(t)>0\}}\theta(t)\right\}\\ & +V\mathbb{E}\{E(t)\Theta(t)\}\\ \text{s.t.}:\quad & (2),\ (3),\ \text{and}\ (8). \end{aligned} \quad (21)$$

Here, the constraint in (10) and (11) have been incorporated into the objective (21) through the Lyapunov optimization technique. Specifically, by omitting the constant terms of $\mathcal{P}_2$, we can obtain

$$\min:\ VE(t)-\sum_{i=1}^{N} Q_i(t)B_i(t)-\sum_{i=1}^{N} H_i(t)B_i(t). \quad (22)$$

By rearranging (22), we have the following online optimization problem:

$$(\boldsymbol{\mathcal{P}_3})\quad \min:\ \sum_{i=1}^{N}\left[VP_i-(Q_i(t)+H_i(t))\,r_i(t)\right]\cdot l_i(t). \quad (23)$$

The problem $\mathcal{P}_3$ is now converted into the linear relaxation of a knapsack problem $\mathcal{P}_4$ as

$$\begin{aligned}(\boldsymbol{\mathcal{P}_4})\quad \max:\quad & \sum_{i=1}^{N} U_i(t)\cdot l_i(t)\\ \text{s.t.}:\quad & (2),\ (3),\ \text{and}\ (8)\end{aligned} \quad (24)$$

where $U_i(t)=(Q_i(t)+H_i(t))r_i(t)-VP_i$. To put it vividly, how to put some "object i" whose "value" is $U_i(t)$ and "weight" is $l_i(t)$ into the "backpack" according to the rules so that the total "value" of the backpack highest. And the maximum load bearing of the backpack is $K(t)\tau_0$. The DAEE algorithm we designed greedily chooses to put "high-value objects" into the "backpack" until the objects overflow. Combining the aforementioned content, DAEE will execute offloading action in the following two scenarios.

1) If the wireless network quality is good enough, DEAA will offload as many workloads as possible to save energy.
2) If there are unstable queues in the buffer, the workloads will be forced to offload in order not to violate the deadline.

We sort the "value" of the "objects" in descending order. The offloading action stops when the "backpack" overflows, which is the first termination condition given by

$$\delta_1=\arg\min_t \sum_{j=1}^{i} T_j(t) > K(t)\cdot\tau_0, \quad (25)$$

where the "value" of $U_i(t)$ should be nonnegative, so the second termination condition δ_2 can be identified, as follows:

$$\delta_2=\arg\min_t U_i(t)<0. \quad (26)$$

Therefore, the final termination condition δ can be tightened to $\delta=\min\{\delta_1,\delta_2\}$. And the corresponding optimal offloading duration on each time slot can be given by

$$l_i^*=\begin{cases} T_i(t), & \text{if } i<\delta\\ K(t)\tau_0-\sum_{i=1}^{\delta-1}T_i(t), & \text{if } i=\delta\\ 0, & \text{if } i>\delta. \end{cases} \quad (27)$$

Specifically, the DAEE algorithmic process is described in Algorithm 1 in detail.

Algorithm 1: The DAEE offloading algorithm.

Input: The parameter V, ϵ_i for each application $i\in\mathcal{N}$
Output: Actual queue $Q(t)$, virtual queue $H(t)$, the offloading action l^*

```
Initialize V and ε_i, where i ∈ N
forall time slots t ∈ T do
    forall i = 1 to N do
        Acquire A_i(t), Q_i(t) and H_i(t)
        Calculate r_i(t), T_i(t) and U_i(t)
    end
    Rank the applications according to U_i(t) in
     descending order with ranked set N';
    Calculate the breaking item δ
    forall j ∈ N' do
        Set the offloading decision l_j^* according to (27);
        Update the actual queue Q_j(t) with (10);
        Update the virtual queue H_j(t) with (11);
    end
end
```

C. Performance Bounds

In this subsection, we focus on the average energy consumption and traffic delay. According to Little's Law [29], the average traffic delay experienced by the MEC server is proportional to the average number of unexecuted workloads, which is the average sum queue length of the backlog. Thus, the average traffic delay can be calculated as follows:

$$\bar{Z}=\frac{\sum_{i\in\mathcal{N}}\limsup_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[Q_i(t)\right]}{\sum_{i\in\mathcal{N}}\lambda_i}. \quad (28)$$

Lemma 3: If the original problem $\mathcal{P}$ has a solution, then there must be an optimal π-only policy, which is independent of the queue backlog, and makes the offloading action $l(t)$ follow the fixed probability distribution. For any task arrival rate λ that is strictly within the network capacity region Λ. There exists a randomized control stationary policy π that yields the following steady-state values:

$$\mathbb{E}\left\{E^{\pi}(t)\right\} = E^{\text{opt}}(\lambda) \tag{29}$$

$$\mathbb{E}\left\{A_i(t)\right\} \leq \mathbb{E}\left\{r_i(t) l_i^{\pi}(t)\right\} \tag{30}$$

where $E^{\text{opt}}(\lambda)$ represents the optimal energy consumption with the task arrival rate λ.

The proof process of Lemma 3 here is similar to the method used in [28], we omit the details for brevity. Then, by applying Lemma 3, we further derive Theorem 1, which states that both average energy consumption and traffic delay can be controlled by corresponding bounds.

Theorem 1: Assuming that there exists a positive α that the data arrival rate $\lambda + \alpha$ is strictly within the network capacity region Λ. Then, under Algorithm 1, for the given V, the performance bounds of the time average energy consumption and traffic delay can be denoted as

$$\bar{E} \leq \frac{D}{V} + E^{\text{opt}} \tag{31}$$

$$\bar{Z} \leq \frac{D + V \cdot E^{\text{opt}}}{\alpha \sum_{i \in \mathcal{N}} \lambda_i}. \tag{32}$$

The detailed proof is deferred to Appendix C.

D. Ensuring Bounded Worst-Case Delay

We can also theoretically get the worst-case delay of each device, and the specific details are given by the following theorem.

Theorem 2: Any algorithm that can maintain bounded $Q_i(t)$ and $H_i(t)$ can also ensure a bounded worst-case delay, that is, $Q_i(t) \leq Q_i^{\max}$ and $H_i(t) \leq H_i^{\max}$, $t \in 0, 1, \ldots, T-1$, the worst-case delay of non-dropped data in queue i is bounded by the constant $T_i^{\max}$ defined as follows [27]:

$$T_i^{\max} = \left\lceil \frac{Q_i^{\max} + H_i^{\max}}{\varepsilon_i} \right\rceil \tag{33}$$

where $\lceil x \rceil$ denotes the smallest integer that is greater than or equal to x.

The detailed proof is deferred to Appendix D. Therefore, when we strictly keep $Q_i(t)$ and $H_i(t)$ within the appropriate upper bounds $Q_i^{\max}$ and $H_i^{\max}$, respectively, we can further control the deadline $T_i^{\max}$.

IV. Performance Evaluation

In this section, we evaluate the performance of DAEE from multiple aspects, and compare it with other workload offloading schemes from multiple perspectives.

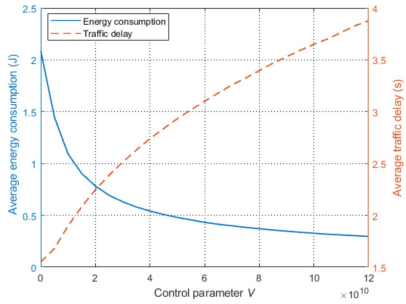

Fig. 3. Average energy consumption and traffic delay.

A. Parameter Settings

The parameter settings for the simulation experiments mainly come from [26] and [30]. We set the number of devices $N = 300$ and the transmit power of each device $P_i \sim U[10, 200]$ mW. A certain area of $\mathcal{X} = \mathcal{Y} = 250$ m that can be marked. Meanwhile, we assume the number of available subchannels $K(t) \sim U[10, 30]$. We choose the Poisson Process to simulate the DSCI-type workloads arrival with $\mathbb{E}\{A_i(t)\} = \lambda_i = \lambda$ for all $i \in \mathcal{N}$, where λ is chosen from $\{3000, 4000, 5000, 6000, 7000, 8000\}$. Besides, the small-scale fading channel power gains follow an exponential distribution, with the unit mean, i.e., $\gamma_i(t) \sim \text{Exp}(1)$. $W = 1$ MHz and $\sigma = 10^{-13}$ W. The pass-loss exponent $\theta = 4$ and the path-loss constant $g_0 = -40$ dB. To get more general experimental results and avoid the uncertainty of wireless network and workload generation, we simulate the system state of 3600 constant time slots with slot length $\tau_0 = 1$ s.

B. Simulation Results

1) Energy-Delay Tradeoff: As depicted in Fig. 3, given the arrival rate $\lambda = 5000$ bit/s, as V arises from 1 to 12×10^{10}, the average energy consumption drops from 2.09 to 0.29 J, and the average traffic delay grows from near 1.55 to 3.87 s. Note that energy consumption decreases rapidly at the beginning, and then, tends to descend slowly, while the traffic delay grows linearly with V. The DAEE algorithm gradually converges to the lowest energy consumption level as the V continues to increase. This also quantitatively shows that there is a tradeoff relationship $[O(1/V), O(V)]$ between average energy consumption and traffic delay in (31) and (32).

2) Effect of the Number of IIoT Devices: Fig. 4 illustrates the impact of the number of IIoT devices on energy consumption and queue length, where the number of devices ranges from 200 to 400. It can be observed that there is a positive correlation between the equipment quantity and energy consumption. The relationship between the number of devices and the total queue length is also the same. This is because the increase in the number of IIoT devices causes more workloads to be generated. On the one hand, offloading more workloads will lead to an increase in energy consumption. On the other hand, due to the limited bandwidth of the available channels, only part of the increased workloads can be undertaken, and the remaining part will stay in the buffer and the queue length will increase.

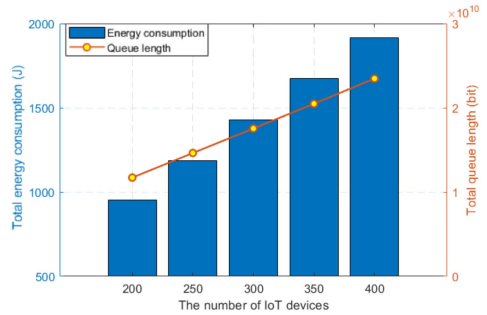

Fig. 4. Impact of the number of IIoT devices on energy consumption and queue length.

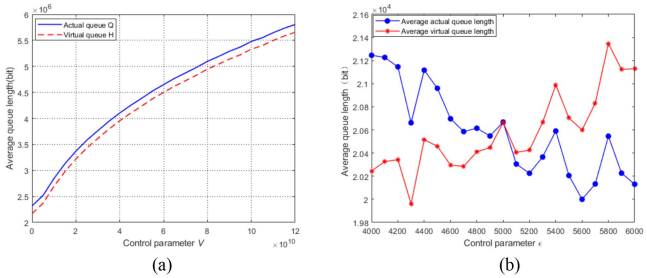

Fig. 5. Average queue backlog of $Q_i(t)$ and $H_i(t)$ with V and ε, respectively. (a) Queue backlog with V. (b) Queue backlog with ε.

3) Actual Queue $Q(t)$ and Virtual Queue $H(t)$: Constructed ε-persistent transmission queue (virtual queue) grows only when there are workloads in the buffer of the actual queue that have not been offloaded for a long time. As shown in Fig. 5(a), with the rise of V, the time-averaged queue backlog of both $Q_i(t)$ and $H_i(t)$ increase. This is because the larger V is, the more important energy consumption is emphasized, and the lower energy consumption is achieved at the expense of a longer queue length. Due to $0 \leq \varepsilon_i \leq \mathbb{E}\{A_i(t)\} = \lambda$, the average queue length of $Q(t)$ is longer than $H(t)$.

In each $H_i(t)$, ε_i plays a vital role in tuning the backlog of $H_i(t)$ during the virtual arrival process, thereby, affecting the online offloading duration. We found that the value of ε_i significantly affects the stability of the queues, as shown in Fig. 5(b). The larger value of ε_i is, the faster the average backlog of $H_i(t)$ rises. This leads to a substantial increase in the "value" $U_i(t)$ of the device i. A higher $U_i(t)$ will strive for a longer offloading duration, which in turn leads to more energy consumption. Thus, the backlog of $Q_i(t)$ falls because of the energy-delay tradeoff.

4) Effect of the Workload Arrival Rate λ and Parameter V: Fig. 6 presents the fluctuation of the actual queue backlog at different λ and V. In Fig. 6(a), as λ becomes larger, the queue length also increases. This is because as λ increases, so do the workloads. Similarly, Fig. 6(b) illustrates the trend that the queue length increases with the increase of V. This is because, with the increase of V, DAEE will pay more attention to energy saving and weaken the control of workload delay.

Obviously, under different λ and V, DAEE can make the queue length converge quickly in a short time and remain stable. We can also observe that the actual queue backlog fluctuates

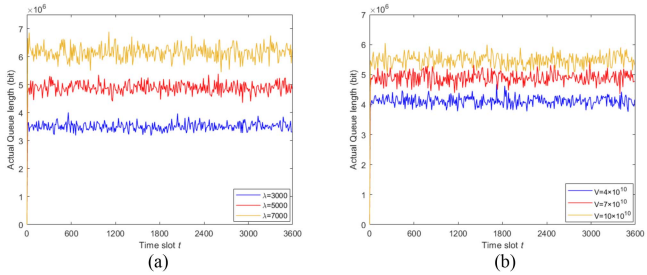

Fig. 6. Effect of the workload arrival rate λ and control parameter V in real-time actual queue backlog. (a) Queue backlog with λ. (b) Queue backlog with V.

frequently, which is the result of DAEE's repeated energy-delay tradeoff on the time slots. When the DAEE algorithm detects that the queue backlog is too large and the deadline may be violated, it will adaptively offload tasks to reduce the queue backlog. When the system's queue backlog is small, the DAEE algorithm will highlight the energy-saving advantage and greedily select a better period to offload tasks, resulting in a gradual increase in the queue backlog.

C. Comparison of Different Offloading Schemes

In this section, we will compare the existing algorithm, baseline algorithms, and modified baseline algorithms to more fully reflect the performance of DAEE.

1) *Energy-efficient dynamic offloading algorithm (EEDOA):* This method is a workload distribution strategy based on Lyapunov optimization [20].
2) *Longest queue length first (LQLF):* In this method, the offloading duration is allocated according to its queue length. Particularly, the queue length is longer, and the corresponding IIoT device has a higher priority to be served.
3) *Longest queue length first with mobility (LQLF-mobility):* The mobility of IIoT devices is considered based on the LQLF greedy algorithm.
4) *Equal opportunity allocation (EOA):* In this method, each device is served by equal opportunity, that is, the available subchannels are equally allocated to each device to the greatest extent.
5) *Equal opportunity allocation with mobility (EOA-mobility):* The mobility of IIoT devices is considered based on the EOA baseline algorithm.

1) Queue Backlog and Energy Consumption: Fig. 7 shows the queue backlog and energy consumption of different offloading schemes in each time slot. It can be seen that under the condition of the same arrival rate, the energy consumption of DAEE is very small, and its performance is much better than LQLF and EOA. When $\lambda = 3000$ bit/s, although the queue length maintained by DAEE is higher than the EOA algorithm that greedily offloads the workload of every device, it is approximately the same as the average queue length of LQLF. Simultaneously, it can save about 87% of energy consumption. When $\lambda = 6000$ bit/s, compared with EOA, our proposed DAEE scheme with perturbed Lyapunov optimization can help to save

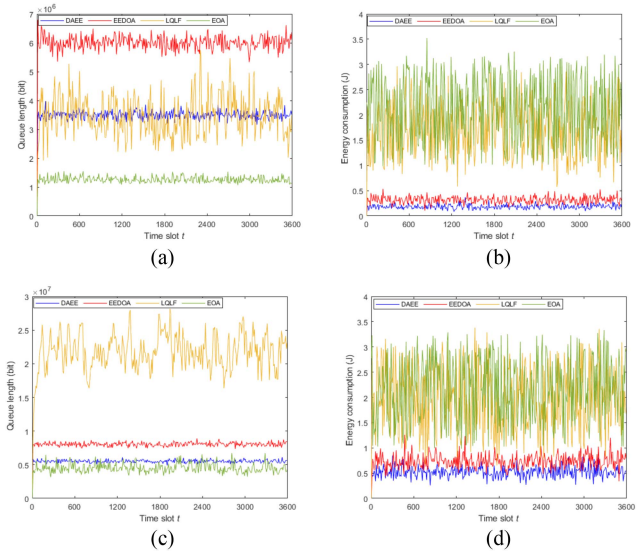

Fig. 7. Different offloading schemes of the queue backlog and energy consumption. (a) $\lambda = 3000$ bit/s. (b) $\lambda = 3000$ bit/s. (c) $\lambda = 6000$ bit/s. (d) $\lambda = 6000$ bit/s.

TABLE I
SYSTEM THROUGHPUT (GBIT)

Throughput / λ(bit/s) / Scheme	3000	4000	5000	6000	7000	8000
DAEE	3.24	4.32	5.39	6.47	7.55	8.63
EEDOA	3.23	4.31	5.39	6.46	7.53	8.58
LQLF	3.23	4.31	5.39	6.45	7.53	8.59
EOA	9.75	9.70	9.71	9.73	9.73	9.72

around 75% of energy consumption, while sacrificing only a small amount of latency.

In addition, it can be seen from Fig. 7(a) and (c) that by traversing all time slots, DAEE and EEDOA can effectively stabilize the queue backlog at a certain level. Since EOA always chooses to greedily offload the workload of all devices, it can also maintain a low level of queue length, while LQLF has a poor ability to control queue stability. With the increase of λ, the queue backlog of LQLF becomes unusually large and unstable. And a large queue length indicates a longer delay. As expected, applying Lyapunov optimization can effectively maintain the stability of the queue backload in the buffer.

2) Throughput: It can be seen from Table I that the throughput of the system and the workload arrival rate λ are in a positive correlation. EOA enables the system to have higher and more stable throughput, and the throughput of other algorithms is roughly the same. This is because EOA does not consider whether the channel conditions are good or bad, and offload tasks within the capacity to the greatest extent. Meanwhile, when λ is large, DAEE makes the system throughput slightly higher than EEDOA and LQLF by cleverly using the role of virtual queues.

3) Average Queue Length and Average Energy Consumption: In order to observe the performance of various algorithms under different arrival rates, we select the average energy consumption and the average queue length (when the arrival rate is fixed, the trend of the average traffic delay and average queue length is consistent) to reflect their ability to save energy and control delay, as shown in Fig. 8. It can be observed that with the increase of λ, the average energy consumption of all algorithms except EOA and EOA-mobility increases. This is because the increase in λ causes more workloads to be generated. Offloading more workloads causes an increase in energy consumption. However, EOA and EOA-mobility insist on offloading all workloads greedily, no matter whether the network connection is good or bad, this is an ideal way to reduce latency, but it has the highest energy consumption. In addition, it can be seen from the figure that due to the increased mobility of the devices, the offloading rate changes, making LQLF-mobility consume more energy than LQLF. As bandwidth resources are limited, this gap will gradually narrow. In contrast, our proposed DAEE has the best energy-saving effect.

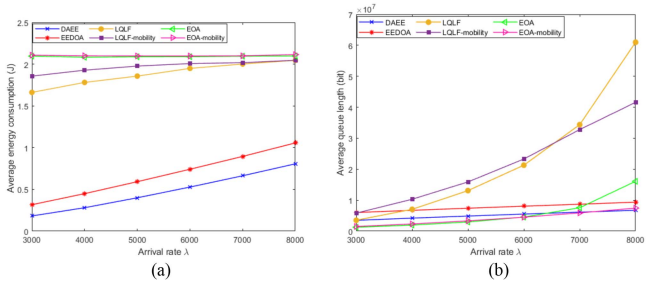

Fig. 8. Average energy consumption and average queue length of different offloading schemes along with λ (bit/s). (a) Average energy consumption. (b) Average queue length.

Fig. 8(b) shows that as λ increases, the average queue length of all algorithms increases. When λ is small, LQLF and LQLF-mobility maintain a low queue length, but they have a very large queue backlog when the arrival rate is large, so they cannot be applied to DSCI-type tasks well. Meanwhile, we found that the average queue length of DAEE and EEDOA grew slowly, which also proved the advantage of Lyapunov optimization in system stability. Here, LQLF-mobility and EOA-mobility have a larger turning point, because some devices may move to a state with better network conditions, which can transmit more workloads, and this gap is more obvious when the λ is larger. By contrast, the effect of DAEE in controlling delay is close to EOA and EOA-mobility, especially when λ is large. In conclusion, the simulation results confirmed the effectiveness of DAEE in saving energy and maintaining lower latency for IIoT-edge systems, and it is more suitable for DSCI-type tasks.

V. CONCLUSION

In this article, we took the mobility of IIoT devices into consideration and proposed an energy-saving workload offloading algorithm, DAEE, with delay awareness. The DAEE used perturbed Lyapunov optimization to construct virtual queues to assist actual queues in optimizing workload distribution. And it can adaptively offload tasks according to the current queue backlog and channel conditions without any future information as a priority. Relevant simulation experiments showed that our framework can adjust the relative importance between the

average traffic delay and the long-term energy consumption by controlling the parameter V, which facilitated the quantification of the performance of $[O(1/V), O(V)]$. By comparing it with other offloading schemes, we found that DAEE can effectively reduce long-term energy consumption and maintain a lower delay strongly. Accordingly, it can be well applied to DSCI-type tasks.

In the future, we will consider IIoT-MEC systems with multiple edge servers with limited computing resources, and design link selection for it at the same time. In other words, to reduce the transmission delay and avoid deadline violations, offloading will only be postponed when both WLAN and cellular networks are in bad connectivity. In addition, extending the IIoT-MEC system to the IIoT-edge-cloud system is also part of our future work.

APPENDIX A
PROOF OF LEMMA 1

Proof: Due to $a^2 \leq (a_3 - a_2)^2 + a_1^2$, we have $a^2 \leq a_3^2 + a_2^2 + a_1^2 - 2a_2a_3$.

Since a_1 and a_3 are nonnegative, we know that $a_1a_3 \geq 0$. So, we can get $a^2 \leq a_3^2 + a_2^2 + a_1^2 - 2a_2a_3 + 2a_1a_3$. ■

APPENDIX B
PROOF OF LEMMA 2

Proof: By scaling the one-step conditional Lyapunov drift $\Delta(\Theta(t))$, we get (19).

Then, by adding the energy consumption weighting term $V\mathbb{E}\{E(t) \mid \Theta(t)\}$ to both sides of (19), the result of *Lemma 2* can be obtained. ■

APPENDIX C
PROOF OF THEOREM 1

Proof: According to Lemma 3, for the arrival rate $\lambda + \alpha$, corresponding to an offloading policy π', which satisfies

$$\mathbb{E}\left\{E^{\pi'}(t)\right\} = E^{\text{opt}}(\lambda + \alpha) \tag{34}$$

$$\mathbb{E}\left\{A_i(t) + \alpha\right\} \leq \mathbb{E}\left\{r_i(t) l_i^{\pi'}(t)\right\}. \tag{35}$$

Substituting (34) and (35) into (20), we obtain

$$\begin{aligned}\Delta(\Theta(t)) + V\mathbb{E}\{E(t) \mid \Theta(t)\} &\leq D + V\mathbb{E}\left\{E^{\pi'}(t)\right\} \\ &- \sum_{i=1}^{N} Q_i(t)\mathbb{E}\left\{r_i(t) l_i^{\pi'}(t) - A_i(t) \mid \Theta(t)\right\} \\ &- \sum_{i=1}^{N} H_i(t)\mathbb{E}\left\{r_i(t) l_i^{\pi'}(t) - \varepsilon_i 1_{\{Q_i(t)>0\}} \mid \Theta(t)\right\}. \end{aligned} \tag{36}$$

Since $\varepsilon_i \leq \mathbb{E}\{A_i(t)\} \leq \mathbb{E}\{r_i(t) l_i^{\pi'}(t)\}$ holds all the time, we have

$$\mathbb{E}\left\{r_i(t) l_i^{\pi'}(t) - A_i(t) \mid \Theta(t)\right\} \geq 0 \tag{37}$$

$$\mathbb{E}\left\{r_i(t) l_i^{\pi'}(t) - \varepsilon_i 1_{\{Q_i(t)>0\}} \mid \Theta(t)\right\} \geq 0. \tag{38}$$

Then, we can obtain

$$\Delta(\Theta(t)) + V\mathbb{E}\{E(t) \mid \Theta(t)\} \leq D + V\mathbb{E}\left\{E^{\pi'}(t)\right\}. \tag{39}$$

Summing (39) over all time slots, it holds

$$\begin{aligned}\mathbb{E}\{L(\Theta(T)) - L(\Theta(0))\} + V\sum_{t=0}^{T-1}\mathbb{E}[E(t)] \\ \leq TD + TV\mathbb{E}\left\{E^{\pi'}(t)\right\}. \end{aligned} \tag{40}$$

Because the initial queue states are empty, we have $\mathbb{E}\{L(\Theta(T))\} = 0$ and $\{L(\Theta(0))\} \geq 0$. Thus,

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[E(t)] \leq \frac{D}{V} + \mathbb{E}\left\{E^{\pi'}(t)\right\} \tag{41}$$

let $T \to \infty$ and $\alpha \to 0$, due to Lebesgues dominated convergence theorem, we can obtain the (31).

Recall (35), it holds

$$\mathbb{E}\left\{B_i^{\pi'}(t)\right\} - \mathbb{E}\left\{A_i(t)\right\} \geq \alpha. \tag{42}$$

Thus, combined with (39), (36) can be scaled to

$$\begin{aligned}\Delta(\Theta(t)) + V\mathbb{E}\{E(t) \mid \Theta(t)\} &\leq D + V\mathbb{E}\left\{E^{\pi'}(t)\right\} \\ &- \sum_{i=1}^{N} Q_i(t)\mathbb{E}\left\{r_i(t) l_i^{\pi'}(t) - A_i(t) \mid \Theta(t)\right\} \\ &\leq D + V\mathbb{E}\left\{E^{\pi'}(t)\right\} - \alpha\sum_{i=1}^{N} Q_i(t). \end{aligned} \tag{43}$$

Reorganizing the aforementioned formula, we can obtain

$$\frac{1}{T}\sum_{t=0}^{T-1}\sum_{i=1}^{N} Q_i(t) \leq \frac{D + V\mathbb{E}\left\{E^{\pi'}(t)\right\}}{\alpha}. \tag{44}$$

Let $T \to \infty$ again, we can obtain

$$\bar{Q} \leq \frac{D + V \cdot E^{\text{opt}}}{\alpha}. \tag{45}$$

Finally, by dividing $\sum_{i\in\mathcal{N}} \lambda_i$ at the same time, it further yields (32). ■

APPENDIX D
PROOF OF THEOREM 2

Proof: If the task is not completed within $T_i^{\max}$ slots, then $Q_i(\tau) > 0\ \forall \tau \in \{t+1, t+2, \ldots, t+T_i^{\max}\}$. It follows from (6) and (7) that for all $\tau \in \{t+1, t+2, \ldots, t+T_i^{\max}\}$, and we have

$$\begin{aligned}H_i(\tau+1) &= \max\{H_i(\tau) - B_i(\tau) + \varepsilon_i, 0\} \\ &\geq H_i(\tau) - B_i(\tau) + \varepsilon_i. \end{aligned} \tag{46}$$

Summing the aforementioned over $\tau \in \{t+1, t+2, \ldots, t+T_i^{\max}\}$, we can obtain

$$H_i\left(\tau + T_i^{\max} + 1\right) - H_i(\tau+1) \geq \varepsilon_i T_i^{\max} - \sum_{\tau=t+1} B_i(\tau). \tag{47}$$

Reorganizing the aforementioned formula, we can obtain

$$\sum_{\tau=t+1}^{t+T_i^{\max}} B_i(\tau) \geq \varepsilon_i T_i^{\max} - H_i^{\max} \tag{48}$$

according to the first-in-first-out data processing method, when the last of the $A_i(\tau)$ data departs in the time slot $t + T_i^{\max}$, it holds $\sum_{\tau=t+1}^{t+T_i^{\max}} B_i(\tau) \geq Q_i(\tau + 1)$. According to our assumption, it must have

$$\sum_{\tau=t+1}^{t+T_i^{\max}} B_i(\tau) < Q_i(\tau + 1) \leq Q_i^{\max}. \tag{49}$$

Combining (48) and (49) yields

$$\varepsilon_i \cdot T_i^{\max} - H_i^{\max} < Q_i^{\max}. \tag{50}$$

Finally, we have

$$T_i^{\max} < \frac{Q_i^{\max} + H_i^{\max}}{\varepsilon_i}. \tag{51}$$

The result contradicts the definition of $T_i^{\max}$ given in (33), where $\bar{H} \leq \bar{Q}$ and $H_i(\tau)$ is bounded as well as $Q_i(\tau)$. ■

REFERENCES

[1] K. Mekki, E. BAJIC, F. Chaxel, and F. Meyer, "A comparative study of LPWAN technologies for large-scale IoT deployment," *ICT Exp.*, vol. 5, no. 1, pp. 1–7, Mar. 2019.

[2] L. Chettri and R. Bera, "A comprehensive survey on Internet of Things (IoT) toward 5G wireless systems," *IEEE Internet Things J.*, vol. 7, no. 1, pp. 16–32, Jan. 2020.

[3] J. Chen, H. Wu, R. Li, and P. Jiao, "Green-parallel online offloading for DSCI-type tasks in IoT-edge systems," *IEEE Trans. Ind. Inform.*, to be published, doi: 10.1109/TII.2022.3167668.

[4] X. Xu et al., "Edge server quantification and placement for offloading social media services in industrial cognitive IoV," *IEEE Trans. Ind. Inform.*, vol. 17, no. 4, pp. 2910–2918, Apr. 2021.

[5] L. Zhang and N. Ansari, "Latency-aware IoT service provisioning in UAV-aided mobile-edge computing networks," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 10573–10580, Oct. 2020.

[6] H. R. Chi, M. F. Domingues, and A. Radwan, "Complex network analysis for ultra-large-scale MEC small-cell based peer-offloading," in *Proc. IEEE Glob. Commun. Conf.*, 2021, pp. 1–6.

[7] P. Brown, "Is battery life hindering the growth of Internet of Things devices," Nov. 05, 2018. [Online]. Available: https://electronics360.globalspec.com/article/13112/is-battery-life-hindering-the-growth-of-internet-of-things-devices

[8] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proc. IEEE*, vol. 107, no. 8, pp. 1738–1762, Aug. 2019.

[9] G. Zhang, W. Zhang, Y. Cao, D. Li, and L. Wang, "Energy-delay tradeoff for dynamic offloading in mobile-edge computing system with energy harvesting devices," *IEEE Trans. Ind. Inform.*, vol. 14, no. 10, pp. 4642–4655, Oct. 2018.

[10] G. Qu, N. Cui, H. Wu, R. Li, and Y. Ding, "ChainFL: A simulation platform for joint federated learning and blockchain in edge/cloud computing environments," *IEEE Trans. Ind. Inform.*, vol. 18, no. 5, pp. 3572–3581, May 2022.

[11] H. Tang, H. Wu, Y. Zhao, and R. Li, "Joint computation offloading and resource allocation under task-overflowed situations in mobile-edge computing," *IEEE Trans. Netw. Serv. Manage.*, vol. 19, no. 2, pp. 1539–1553, Feb. 2022.

[12] Z. Sheng, C. Mahapatra, V. C. M. Leung, M. Chen, and P. K. Sahu, "Energy efficient cooperative computing in mobile wireless sensor networks," *IEEE Trans. Cloud Comput.*, vol. 6, no. 1, pp. 114–126, Jan. 2018.

[13] H. Wu, Y. Sun, and K. Wolter, "Energy-efficient decision making for mobile cloud offloading," *IEEE Trans. Cloud Comput.*, vol. 8, no. 2, pp. 570–584, Apr.–Jun. 2020.

[14] S. P. Meyn and R. L. Tweedie, *Markov Chains and Stochastic Stability*. Berlin, Germany: Springer, 2012.

[15] H. Wu, K. Wolter, P. Jiao, Y. Deng, Y. Zhao, and M. Xu, "EEDTO: An energy-efficient dynamic task offloading algorithm for blockchain-enabled IoT-edge-cloud orchestrated computing," *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2163–2176, Feb. 2021.

[16] M. J. Neely, "Stability and probability 1 convergence for queueing networks via Lyapunov optimization," *J. Appl. Math.*, vol. 2012, pp. 1–35, 2012.

[17] M. Mukherjee, M. Guo, J. Lloret, R. Iqbal, and Q. Zhang, "Deadline-aware fair scheduling for offloaded tasks in fog computing with inter-fog dependency," *IEEE Commun. Lett.*, vol. 24, no. 2, pp. 307–311, Feb. 2020.

[18] M. Guo, L. Li, and Q. Guan, "Energy-efficient and delay-guaranteed workload allocation in IoT-edge-cloud computing systems," *IEEE Access*, vol. 7, pp. 78685–78697, 2019.

[19] P. Shu et al., "eTime: Energy-efficient transmission between cloud and mobile devices," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 195–199.

[20] Y. Chen, N. Zhang, Y. Zhang, X. Chen, W. Wu, and X. S. Shen, "Energy efficient dynamic offloading in mobile edge computing for Internet of Things," *IEEE Trans. Cloud Comput.*, vol. 9, no. 3, pp. 1050–1060, Jul.–Sep. 2021.

[21] Y. Dai, K. Zhang, S. Maharjan, and Y. Zhang, "Deep reinforcement learning for stochastic computation offloading in digital twin networks," *IEEE Trans. Ind. Inform.*, vol. 17, no. 7, pp. 4968–4977, Jul. 2021.

[22] Z. Zhuang, J. Wang, Q. Qi, J. Liao, and Z. Han, "Adaptive and robust routing with Lyapunov-based deep RL in MEC networks enabled by blockchains," *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2208–2225, Apr. 2021.

[23] N. Salodkar, A. Bhorkar, A. Karandikar, and V. S. Borkar, "An on-line learning algorithm for energy efficient delay constrained scheduling over a fading channel," *IEEE J. Sel. Areas Commun.*, vol. 26, no. 4, pp. 732–742, Apr. 2008.

[24] Y. Deng, Z. Chen, X. Yao, S. Hassan, and A. M. A. Ibrahim, "Parallel offloading in green and sustainable mobile edge computing for delay-constrained IoT system," *IEEE Trans. Veh. Technol.*, vol. 68, no. 12, pp. 12202–12214, Dec. 2019.

[25] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3590–3605, Dec. 2016.

[26] H. Zhao, W. Du, W. Liu, T. Lei, and Q. Lei, "QoE aware and cell capacity enhanced computation offloading for multi-server mobile edge computing systems with energy harvesting devices," in *Proc. IEEE Ubiquitous Intell. Comput.*, 2018, pp. 671–678.

[27] M. J. Neely, "Opportunistic scheduling with worst case delay guarantees in single and multi-hop networks," in *Proc. IEEE INFOCOM*, 2011, pp. 1728–1736.

[28] M. Neely, *Stochastic Network Optimization With Application to Communication and Queueing Systems*. San Rafael, CA, USA: Morgan & Claypool, 2010.

[29] L. Georgiadis, M. J. Neely, and L. Tassiulas, "Resource allocation and cross-layer control in wireless networks," *FNT Netw.*, vol. 1, no. 1, pp. 1–144, 2005.

[30] X. Lyu et al., "Optimal schedule of mobile edge computing for Internet of Things using partial information," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2606–2615, Nov. 2017.

Huaming Wu (Senior Member, IEEE) received the B.E. and M.S. degrees in electrical engineering from the Harbin Institute of Technology, Harbin, China, in 2009 and 2011, respectively, and the Ph.D. degree (Highest Hons.) in computer science from Freie Universität Berlin, Berlin, Germany, in 2015.

He is currently an Associate Professor with the Center for Applied Mathematics, Tianjin University, Tianjin, China. His research interests include wireless networks, edge computing, Internet of Things, and deep learning.

Junqi Chen received the B.S. degree in mathematics and applied mathematics from the Taiyuan University of Technology, Taiyuan, China, in 2019, and the M.S. degree in mathematics from the Center for Applied Mathematics, Tianjin University, Tianjin, China, in 2022.

Her research interests include Internet of Things, deep learning, and edge computing.

Tu N. Nguyen (Senior Member, IEEE) received the Ph.D. degree in electronic engineering from the National Kaohsiung University of Science and Technology (formerly, National Kaohsiung University of Applied Sciences), Kaohsiung, Taiwan, in 2016.

He is currently an Assistant Professor of computer science with Kennesaw State University (KSU), Kennesaw, GA, USA. Prior to joining the KSU, he was an Assistant Professor of computer science with Purdue University Fort Wayne, Fort Wayne, IN, USA. He was a Postdoctoral Associate with the Department of Computer Science and Engineering, University of Minnesota Twin Cities in 2017. Prior to joining the University of Minnesota, he worked with Intelligent Systems Center, Missouri University of Science and Technology as a Postdoctoral Researcher in 2016. He has authored and coauthored more than 100 publications in leading academic journals as well as conferences, including IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT, IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, *ACM Transactions on Internet Technology*, IEEE TRANSACTIONS ON NETWORK SCIENCE AND ENGINEERING, IEEE TRANSACTIONS ON COGNITIVE COMMUNICATIONS AND NETWORKING, *ACM Transactions on Sensor Networks*, IEEE TRANSACTIONS ON POWER DELIVERY, and IEEE Conference on Local Computer Networks (LCN). His research interests include developing fundamental mathematical tools and principles to design and develop smart, secure, and self-organizing systems, with applications to network systems, cyber-physical systems, quantum networks, and quantum computing.

Dr. Nguyen has been involved in many professional activities, including serving as Editor/Guest Editor for academic journals such as an Associate Editor for IEEE SYSTEMS JOURNAL, *Journal of Combinatorial Optimization*, and IEEE ACCESS; a leading Guest Editor for IEEE TRANSACTION ON COMPUTATIONAL SOCIAL SYSTEMS, IEEE INTERNET OF THINGS MAGAZINE, and IEEE JOURNAL OF BIOMEDICAL AND HEALTH INFORMATICS; and a Technical Editor for *Computer Communication*. He is the Co-Editor-in-Chief of the book series: *IET Advances in Distributed Computing* and *Block-Chain Technologies*. He has been in different organizing committees such as being Technical Program Committee (TPC) chairs and general chairs for several IEEE/ACM/Springer flagship conferences. He has also served as a TPC member for several international conferences including IEEE INFOCOM, IEEE GLOBECOM, IEEE LCN, IEEE International Conference on RFID, IEEE International Conference on Communications, IEEE Wireless Communications and Networking Conference, and so on. He was the recipient of an US NSF CRII Award in 2021.

Huijun Tang received the B.Sc. degree in information and computing science from Jinan University, Guangzhou, China, in 2016, and the M.S. degree in applied mathematics and the Ph.D. degree in mathematics from Tianjin University, Tianjin, China in 2018 and 2015, respectively.

She is currently a Lecture with the School of Cyberspace, Hangzhou Dianzi University, Hangzhou, China. Her research interests include Internet of Things, mobile edge computing, and deep learning.