Hindawi
Mobile Information Systems
Volume 2022, Article ID 1750192, 17 pages
https://doi.org/10.1155/2022/1750192

Research Article

Multitask Offloading Strategy for Edge Computing in Time-Sensitive Networking

Junhua Chen,[1,2] **Chenggen Pu**,[1,2] **Ping Wang**,[2] **Xueda Huang**,[2] **Yan Zhang**,[2] **and Yanfei Liu**[3]

[1]*School of Computer Science and Technology, Chongqing University of Posts and Telecommunications, Chongqing 400065, China*
[2]*Key Laboratory of Industrial Internet of Things & Networked Control, Chongqing University of Posts and Telecommunications, Chongqing 400065, China*
[3]*School of Artificial Intelligence, Chongqing University of Technology, Chongqing 400054, China*

Correspondence should be addressed to Yanfei Liu; liuyanf@cqut.edu.cn

Received 13 October 2022; Revised 18 December 2022; Accepted 21 December 2022; Published 30 December 2022

Academic Editor: Sebastian Podda

Edge computing is a creative computing paradigm that enhances the computing capacity of the edge device close to the data source. As the key technology of edge computing, task offloading, which can improve the response speed and the stability of the network system, has attracted much attention and has been applied in many network scenarios. However, few studies have considered the application of task offloading in time-sensitive networking (TSN), which is a promising technology that has the potential to guarantee data delivery with bounded latency and low jitter. To this end, we establish a task offloading stream transmission model for TSN based on the queueing theory. With the model, the average response time can be achieved by quantitative calculation. Then, we introduce the backward method to construct a utility function and formulate an exact potential game to model the task offloading competition among edge devices considering the minimization of the average response time of all tasks. Furthermore, a distributed and sequential decision-making algorithm for multitask offloading (DSDA-MO) is proposed to find the Nash equilibrium. Through numerical studies, we evaluate the algorithm performance as well as the benefit of the multitask offloading mechanism. The results reveal that through the proposed game theoretic approach, we can obtain the optimal multitask offloading strategy, which can significantly reduce the task computation delay in TSN, within a finite number of rounds of calculation.

1. Introduction

With the advent of the era of Industry 4.0, related technologies, such as artificial intelligence (AI), cyber-physical systems (CPSs), and industrial internet of things (IIOT), have been deeply studied and developed [1, 2]. In many cases, such as the case of industrial equipment and systems sharing the communication channel in the industrial field network, the data packet delivery requires bounded latency and low jitter, which are known as deterministic real-time communication. The traditional network technology can no longer meet the requirements of deterministic real-time transmission of industrial data, let alone support the mixed transmission of periodic control data and emergency data in many complex industrial scenarios. To overcome this challenge, IEEE promotes a new real-time communication protocol standard, IEEE 802.1 Time-sensitive networking (TSN) [3, 4], which evolved based on IEEE 802.1 audio video bridging (AVB) series standards. Utilizing network-wide clock synchronization, time-aware haper (TAS), traffic scheduling, and other mechanisms, TSN realizes the shared network transmission of time-sensitive streams and nonreal-time streams within the local area network.

During the development process of industrial network technology, the industry and academy find that the traditional centralized cloud computing model cannot satisfy the requirement of real-time computing in the industrial field. Therefore, a new computing paradigm, named edge

computing (EC), has come out to address the challenge of real-time processing data from the network edge. EC provides an architecture for migrating computation power from the remote cloud to the local position close to the data source by deploying edge computing nodes (ECNs) or edge computing servers (ECSs) near the edge devices [5–8].

Since TSN can reduce the delay of data transmission and EC can reduce the time cost of data computation, the combination of TSN and EC technologies will be able to decrease the overall delay of the industrial system. The core principles of EC are computing resource allocation and computing service provision. A common application of EC is task offloading from edge devices to ECN/ECS, i.e., tasks generated in edge devices are delivered to ECN/ECS for remote computing [9]. Recently, several studies have investigated computing task offloading strategies in many network environments for EC [10–12]. However, to the best of our knowledge, few studies have considered their applications in TSN. We are the first to study the computing task offloading strategy in TSN.

The TSN standard comprises a series of protocol clusters, of which IEEE 802.1Qbv [13] is now widely applied in industrial scenarios. The core of IEEE 802.1Qbv is the principle of time-triggered communication controlled by gate control list (GCL) according to the traffic priority. The GCL is usually generated in advance of data delivery in TSN according to the transmission requirements of the given application. Generally, edge devices are ignorant of the details of GCL. Hence, it is challenging to evaluate the task offloading cost and design the multi-task offloading strategy, i.e., whether each task should be computed locally or offloaded to an ECN/ECS. For scenarios that are similar to the uncertain transmission conditions, Li [14] applied the queuing theory to model and evaluate the average delay encountered in task offloading. Moreover, Li proposed a strategy for adjusting the arrival rates of tasks offloaded from user equipment (UE) to different mobile edge computing (MEC) [15] servers based on the average delay. Motivated by these ideas, this paper treats each edge device as an M/G/1 queueing system continuously generating multitasks. Moreover, it is assumed that the hybrid stream, which is composed of the task offloading substreams and the original TSN substreams with the same priority, injects into an M/G/1 queue with server breakdown in the TSN switch. Based on it, this paper proposes a system model and designs a strategy decision-making algorithm on the basis of the potential game theory. The technical contributions of this article can be summarized as follows:

(1) This paper establishes a task offloading streams transmission model by treating each edge device as an M/G/1 queueing system continuously generating multitasks and treating the TSN switch as many M/G/1 queueing systems with server breakdown. Through the proposed model, the average response time of tasks generated on each edge device can be obtained through quantitative calculation, and the optimal multitask offloading strategy of the edge device can be studied mathematically and rigorously.

(2) This paper constructs a utility function with the backward method and formulates an exact potential game to model the task offloading competition among edge devices considering the minimization of the average response time of all tasks.

(3) To effectively find the Nash equilibrium, a distributed and sequential decision-making algorithm for multitask offloading (DSDA-MO) is proposed, and its performance is studied in simulation experiments.

The remainder of this paper is structured as follows: in Section 2, the related work is presented. In Section 3, we describe the system model. In Section 4, we introduce the game formulating the competition among edge devices for edge computing resources and propose a DSDA-MO algorithm to search the Nash equilibrium. In Section 5, numerical results are presented, and the algorithm is evaluated. Finally, the paper is concluded in Section 6.

2. Related Work

As the key technology of EC, computing offloading refers to offloading tasks generated on terminal devices to ECN/ECS based on rational offloading decisions and resource allocation strategies [16]. Utilizing the computing offloading mechanism, the terminal devices, also named edge devices, with limited computing resources can achieve less response time and energy consumption by the tasks executing on ECN/ECS. There are three main goals for task offloading decision-making: shorten the task computing time, reduce the device energy consumption, and jointly optimize the weighted sum of task computing time and energy consumption.

Over the past few years, task offloading in MEC has received more and more attention. Several studies jointly model the computing mode decision problem and resource allocation problem. Liu et al. [17] propose an efficient one-dimensional search algorithm to find the optimal task scheduling strategy. It uses the Markov chain to analyze the average delay and energy consumption of each task on the mobile device for computing task offloading strategies, according to the queuing state of the task buffer, the execution state of the local processing unit, and the state of the transmission unit. It establishes a mathematical model of delay minimization problems with power constraints. Chen and Hao [18] study the task offloading problem in ultradense networks from the perspective of software-defined networking (SDN) thinking and model the task offloading problem as an NP-hard mixed integer nonlinear programming (MINLP) problem. In reference [19], mobile devices offload computing tasks to multiple ECSs and download results from the MEC servers in a preset time slot. By jointly optimizing task scheduling and resource allocation, the computational delay of tasks is minimized.

Recently, many pieces of literature have proposed some approaches to deal with multitask offloading situations [20–23]. In these approaches, the author of reference [20]

constructs a multitask scheduling scheme for multicore mobile devices to balance the execution cost and energy consumption; the author of reference [21] aims to the tradeoff for energy consumption and time cost of a single task with different offloadable components, the author of reference [22] proposes a decision-making strategy for the multitask offloading game and constructs the Nash equilibrium. In addition, many researchers apply game theoretic approaches for computation offloading and resource allocation with minimizing the time cost as an optimal objective, by treating the multiuser computation offloading strategy making as a noncooperative game [24–29].

Overall, a lot of literature focuses on modeling task offloading decision-making mode and has promoted many excellent algorithms to resolve the optimal problem of the offloading strategies for EC in the mobile network. However, few studies have considered the TSN scenarios. To the best of our knowledge, this is the first study that reviews the task offloading problem for EC in the TSN environment. In TSN, the tasks are generated in the queue in each edge device, and the transmission packages are cached in the queues of TSN switches for delivery controlled by GLC. Utilizing queueing theory to evaluate task offloading time cost seems an appropriate choice. The author of reference [30] uses M/M/1 queueing theory to model the task offloading competition as a noncooperative game in a three tiers architecture, consisting of mobile nodes, cloudlets, and cloud servers. Li [14] establishes an M/G/1 queueing model for the UEs and an M/G/m queueing model for the MEC and models a noncooperative game to study the stabilization of a competitive mobile edge computing environment. Inspired by the previous studies, we model multitask offloading competition in TSN as a potential game by using queueing theory.

3. System Model

This section studies a TSN system comprising multiple TSN end nodes, one TSN switch, M edge devices denoted by the set $\mathscr{M} = \{D_1, D_2, \ldots, D_M\}$, and N ECS*s* denoted by the set $\mathscr{N} = \{E_1, E_2, \ldots, E_N\}$. The multitask offloading in the TSN system is shown in Figure 1, and the notations and definitions in the proposed system model are listed in Table 1. We consider that each edge device has a queueing system to process the continuously generated tasks. A task in D_i can be further described as $t_i = (\ell_i, \rho_i)$, where ℓ_i is the task body size (including the code and data) and ρ_i is the processing density in cycles per bit, i.e., the number of cycles required to process a unit bit of the task body. The task processing time in the edge device can be calculated as $(\ell_i * \rho_i)/\widehat{f}_i$, where $\widehat{f}_i$ is the process frequency of the given edge device D_i in cycles per second (e.g., MIPS). Usually, ρ_i is treated as a constant according to the task application type and ℓ_i follows an arbitrary probability distribution; thus, the task computing time is also an arbitrary random variable. We assume that in each edge device, the task interarrival time follows an exponential distribution. Therefore, each edge device can be treated as an M/G/1 queueing system.

Benefiting from the edge computing technology, edge devices could choose tasks to be offloaded to the nearby ECS*s* to achieve faster computation speed or less power consumption than local computing. The tasks on D_i are divided into two types of M/G/1 substreams: unoffloadable computing task substream and offloadable computing task substream. The latter includes local computing tasks and remote computing tasks offloaded to other ECS*s*. Correspondingly, the whole task arrival rate of D_i can be formulated as $\lambda_i = \lambda_{i,\Delta} + \lambda_{i,0} + \sum_{n=1}^{N} \lambda_{i,n}$, where $\lambda_{i,\Delta}$ is the arrival rates of the tasks unoffloadable, $\lambda_{i,0}$ is the arrival rates of the tasks that are offloadable and executed locally, and $\lambda_{i,n}$ is the arrival rates of the tasks that are to be offloaded to the ECS E_n. The arrival rate distribution vector $(\lambda_{i,\Delta}, \lambda_{i,0}, \lambda_{i,1}, \cdots, \lambda_{i,N})$ (for $1 \le i \le M$) represents the computation offloading strategy of D_i. Similarly, there are two types of task body sizes, which are denoted as $\ell_{i,\Delta}$ and $\ell_{i,*}$ corresponding to the aforementioned unoffloadable and offloadable task substreams, respectively; these variables are independent and identically distributed (i.i.d.) random variables with an arbitrary probability distribution. Since most devices generally work on periodic duty in TSN, the statistical distribution of the attributes of the tasks in the edge devices can be obtained via long-term statistics. Herein, we assume that the expected values of $\ell_{i,\Delta}$ and $\ell_{i,*}$, i.e., $E[\ell_{i,\Delta}]$ and $E[\ell_{i,*}]$, and their second moment, $E[\ell_{i,\Delta}^2]$ and $E[\ell_{i,*}^2]$, are available.

Each task generated in an edge device has two choices: local computing or remote computing in one of the ECS*s*. We discuss the two cases of the local computing and the remote computing approaches here.

3.1. Local Computing. The local computing task stream is composed of an unloadable task substream and a loadable local computing task substream; the arrival rates of these substreams are given by $\lambda_{i,\Delta}$ and $\lambda_{i,0}$, respectively. Let $C_{i,0}$ denote the tasks' local computing time in D_i; then, the average time for local computing is as follows:

$$\mathrm{E}\left[C_{i,0}\right] = \frac{\lambda_{i,\Delta}}{\lambda_{i,\Delta} + \lambda_{i,0}} \cdot \frac{E\left[\ell_{i,\Delta}\right] \times \rho_i}{\widehat{f}_i} + \frac{\lambda_{i,0}}{\lambda_{i,\Delta} + \lambda_{i,0}} \cdot \frac{E\left[\ell_{i,*}\right] \times \rho_i}{\widehat{f}_i}. \tag{1}$$

The second moment of $C_{i,0}$ can be obtained as follows:

$$E\left[C_{i,0}^2\right] = \frac{\lambda_{i,\Delta}}{\lambda_{i,\Delta} + \lambda_{i,0}} \cdot \frac{E\left[\ell_{i,\Delta}^2\right] \times \rho_i^2}{\widehat{f}_i^2} + \frac{\lambda_{i,0}}{\lambda_{i,\Delta} + \lambda_{i,0}} \cdot \frac{E\left[\ell_{i,*}^2\right] \times \rho_i^2}{\widehat{f}_i^2}. \tag{2}$$

Furthermore, according to the Pollaczek–Khintchine formula [31], the average waiting time in the queue is given by the following expression:

$$\overline{\mathscr{W}}_{i,0} = \frac{\left(\lambda_{i,\Delta} + \lambda_{i,0}\right) E\left[C_{i,0}^2\right]}{2\left(1 - \left(\lambda_{i,\Delta} + \lambda_{i,0}\right) E\left[C_{i,0}\right]\right)}, \quad \text{s.t } 1 > \left(\lambda_{i,\Delta} + \lambda_{i,0}\right) E\left[C_{i,0}\right], \tag{3}$$

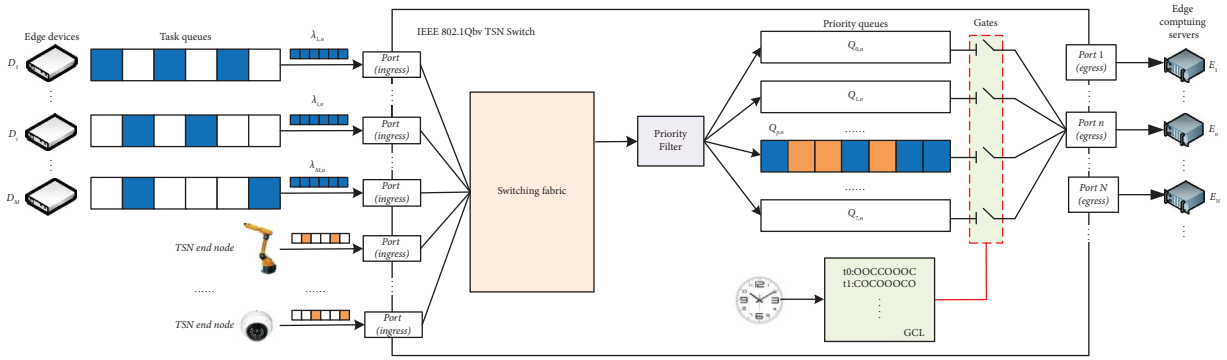

Figure 1: Multitask offloading in time-sensitive networking system. The task offloading substream to ECS n is composed of offloading packages marked in blue from all edge devices, and the original TSN substream to ECS n consists of TSN packages marked in yellow from all TSN end nodes.

Table 1: Summary of notations and definitions.

Notation	Definition
D_i	The edge device i, $1 \le i \le M$
E_n	The ECS n, $1 \le n \le N$
$\mathscr{M}$	The set of edge devices, $\mathscr{M} = \{D_1, D_2, \ldots, D_M\}$
$\mathscr{N}$	The set of ECSs, $\mathscr{N} = \{E_1, E_2, \ldots, E_M\}$
t_i	a task generated on D_i
$\ell_{i,\Delta}$	The size of the unoffloadable tasks on D_i, including code and data
$\ell_{i,*}$	The size of the offloadable tasks on D_i , including code and data
ρ_i	The processing density of tasks in D_i
$\lambda_{i,\Delta}$	The arrival rate of the unoffloadable tasks generated on D_i
$\lambda_{i,n}$	If $n = 0$, the arrival rate of tasks computed locally in D_i if $1 \le n \le N$, the arrival rate of tasks offloaded from D_i to E_n
λ_i	The sum of arrival rates of offloadable tasks generated on D_i, $\lambda_i = \sum_{n=0}^{N} \lambda_{i,n}$
$\widetilde{\lambda}_{p,n}$	The arrival rate of the original TSN stream of priority p to E_n in the TSN gateway
Λ_n	The available maximum offloading task arrival rate from all edge devices to E_n
v	The bandwidth of the TSN network
$C_{i,n}$	If $n = 0$, the execution time of tasks computed locally in D_i if $1 \le n \le N$, the execution time of the tasks generated on D_i and computed remotely in E_n
$\overline{C}_{i,n}$	Mean of $C_{i,n}$
$\overline{Q}_{*,n}$	The average time that the members of the hybrid streams spend in queue $Q_{p,n}$
$\overline{T}_{i,n}$	If $n = 0$, the average time cost of the tasks computed locally in D_i if $1 \le n \le N$, the average time cost of the tasks generated on D_i and computed remotely in E_n
$\overline{T}^{l}_{i,0}$	The average time cost when all tasks are processed locally in D_i
$\overline{T}_i$	The average time cost of all tasks generated on D_i
$\overline{\mathscr{W}}_{*,n}$	The average waiting time of the tasks from all edge devices to E_n spent in TSN gateway
$\mathscr{g}_{p,n}$	The length of the original TSN packages in queue $Q_{p,n}$
h_n	The service interrupted time of queue $Q_{p,n}$
$\Omega_{0,n}$	The service time of the original the TSN substream in $Q_{p,n}$
$\Omega_{*,n}$	All the service time of the compound stream in $Q_{p,n}$
$\widehat{f}_i$	The processing frequency of D_i
f_n	The processing frequency of E_n
$\mathscr{G}$	The multi-task offloading game
$\mathscr{S}_i$	The task offloading strategy of the D_i
$\mathscr{S}_{-i}$	The task offloading strategy set of all edge devices except D_i
$\mathbb{S}$	The task offloading strategy space of all edge devices
ξ	The accuracy parameter

and the average response time for all local computing tasks on D_i is as follows:

$$\begin{aligned}\overline{T}_{i,0} &= \overline{\mathscr{W}}_{i,0} + E\left[C_{i,0}\right] \\ &= \frac{\rho_i^2\left(\lambda_{i,\Delta}E\left[\ell_{i,\Delta}^2\right] + \lambda_{i,0}E\left[\ell_{i,*}^2\right]\right)}{2\widehat{f}_i^2 - 2\widehat{f}_i\rho_i\left(\lambda_{i,\Delta}E\left[\ell_{i,\Delta}\right] + \lambda_{i,0}E\left[\ell_{i,*}\right]\right)} + \frac{\rho_i\left(\lambda_{i,\Delta}E\left[\ell_{i,\Delta}\right] + \lambda_{i,0}E\left[\ell_{i,*}\right]\right)}{\widehat{f}_i\left(\lambda_{i,\Delta} + \lambda_{i,0}\right)}.\end{aligned} \tag{4}$$

If all tasks are executed locally, i.e., $\lambda_{i,0} = \lambda_i - \lambda_{i,\Delta}$, the average response time for all tasks in D_i is as follows:

$$\overline{T}_{i,0}^{l} = \frac{\rho_i^2\left(\lambda_{i,\Delta}\left(E\left[\ell_{i,\Delta}^2\right] - E\left[\ell_{i,*}^2\right]\right) + \lambda_i E\left[\ell_{i,*}^2\right]\right)}{2\widehat{f}_i^2 - 2\widehat{f}_i\rho_i\left(\lambda_{i,\Delta}\left(E\left[\ell_{i,\Delta}\right] - E\left[\ell_{i,*}\right]\right) + \lambda_i E\left[\ell_{i,*}\right]\right)} + \frac{\rho_i\left(\lambda_{i,\Delta}\left(E\left[\ell_{i,\Delta}\right] - E\left[\ell_{i,*}\right]\right) + \lambda_i E\left[\ell_{i,*}\right]\right)}{\widehat{f}_i\lambda_i}. \tag{5}$$

3.2. Remote Computing in ECS. In the scenario of task remote computing, a task generated on one edge device is delivered to an ECS using TSN. To simplify the model, we assume that the tasks generated on edge devices only pass one TSN switch to reach various ECSs and the whole TSN network is consistent with the IEEE 802.1Qbv standard. As shown in Figure 1, each TSN device injects into the TSN data stream from the relevant ingress port of the TSN switch. Then, the switching fabric of the switch redirects the data stream to the proper output port according to the TSN data frame destination. To provide latency and jitter guarantees, before being transported to a specific output port that is connected to E_n, the TSN stream must pass through a priority filter and be reshaped in eight priority queues, $Q_{0,n}, Q_{1,n}, Q_{2,n}, \ldots, Q_{7,n}$, and be finally ejected out from egress port according to the frame priority and GCL state without conflict; this process is known as the time aware shaper (TAS) mechanism. In the computing tasks offloading scenario, the hybrid stream composed of task offloading substreams and original TSN substreams is transmitted over the network. In IEEE802.1Q stand, the stream type and priority are defined in Table 2. To reduce the impact on original TSN data transmission, we set the priority of all the task offloading substreams to be p (e.g., $p = 2$, excellent effort traffic), i.e., all the task offloading streams to E_n get into the queue $Q_{p,n}$, for all $0 \le p \le 7$, $1 \le n \le N$.

First, we ignore the priority queue delivery interruption caused by TAS. Herein, we have three assumptions. First, we assume that before the task offloading substreams were generated, the queue $Q_{p,n}$ was in agreement with the M/G/1 queue model. Second, we assume that the original TSN substream of priority p, whose empress port is the same as the task offloading substreams to E_n, arrives according to a Poisson process with a rate of $\widetilde{\lambda}_{p,n}$, and the length of each package (composed of many TSN frames) is $\mathscr{g}_{p,n}$ following a general distribution. Third, we assume that $E[\mathscr{g}_{p,n}]$ and $E[\mathscr{g}_{p,n}^2]$ are available, and we set the TSN switch egress port sending speed as v. Thus, the service time, $\Omega_{0,n}$, of the original TSN substream in $Q_{p,n}$ is an i.i.d random variable with mean $E[\Omega_{0,n}] = E[\mathscr{g}_{p,n}]/v$ and second moment $E[\Omega_{0,n}^2] = E[\mathscr{g}_{p,n}^2]/v^2$. When the task offloading stream from edge device D_i to E_n arrives at queue $Q_{p,n}$, it is considered as a single Poisson substream with arrival rate $\lambda_{i,n}$. Its service time $\Omega_{i,n}$ is also an i.i.d. random variable with mean $E[\Omega_{i,n}] = E[\ell_{i,*}]/v$ and second moment $E[\Omega_{i,n}^2] = E[\ell_{i,*}^2]/v^2$. The compound stream, composed of the original TSN data Poisson substream and the task offloading Poisson substreams from all devices to E_n, is still a Poisson stream, whose arrival rate is $\widetilde{\lambda}_{p,n} + \sum_{i=1}^{M}\lambda_{i,n}$. Let the service time of each substream in queue $Q_{p,n}$ be denoted by $\Omega_{0,n}, \Omega_{1,n}, \cdots, \Omega_{M,n}$; then, the whole service time $\Omega_{*,n}$ of the compound stream is an i.i.d. random variable with mean

$$\begin{aligned}E\left[\Omega_{*,n}\right] &= \frac{\widetilde{\lambda}_{p,n}E\left[\Omega_{0,n}\right]}{\widetilde{\lambda}_{p,n} + \sum_{i=1}^{M}\lambda_{i,n}} + \frac{\sum_{i=1}^{M}\lambda_{i,n}\Omega_{i,n}}{\widetilde{\lambda}_{p,n} + \sum_{i=1}^{M}\lambda_{i,n}} \\ &= \frac{\widetilde{\lambda}_{p,n}E\left[\mathscr{g}_{p,n}\right] + \sum_{i=1}^{M}\lambda_{i,n}E\left[\ell_{i,*}\right]}{v\left(\widetilde{\lambda}_{p,n} + \sum_{i=1}^{M}\lambda_{i,n}\right)},\end{aligned} \tag{6}$$

and second moment

$$\begin{aligned}E\left[\Omega_{*,n}^2\right] &= \frac{\widetilde{\lambda}_{p,n}E\left[\Omega_{0,n}^2\right]}{\widetilde{\lambda}_{p,n} + \sum_{i=1}^{M}\lambda_{i,n}} + \frac{\sum_{i=1}^{M}\lambda_{i,n}E\left[\Omega_{i,n}^2\right]}{\widetilde{\lambda}_{p,n} + \sum_{i=1}^{M}\lambda_{i,n}} \\ &= \frac{\widetilde{\lambda}_{p,n}E\left[\mathscr{g}_{p,n}^2\right] + \sum_{i=1}^{M}\lambda_{i,n}E\left[\ell_{i,*}^2\right]}{v^2\left(\widetilde{\lambda}_{p,n} + \sum_{i=1}^{M}\lambda_{i,n}\right)}.\end{aligned} \tag{7}$$

According to the standard IEEE802.1Qbv, the transmission of queue $Q_{p,n}$ may be interrupted, i.e., queue service breakdown may occur, due to the GLC state or higher priority queue transmission requirements. We assume that the queue system of $Q_{p,n}$ remains in a stable state before the task offloading substreams reach the switch and that the transmission server breaks down at an exponential rate α_n, i.e., the probability that the queue will be able to serve for an

Table 2: IEEE 802.1Q data stream priority [32].

Priority	Acronym	Stream type
0	BE	Best effort
1	BE	Background
2	EE	Excellent effort
3	CA	Critical applications
4	VI	"Video," <100 ms latency and jitter
5	VO	"Voice," <10 ms latency and jitter
6	IC	Internetwork control
7	NC	Network control

additional time t without breaking down is $e^{-\alpha_n t}$. After the transmission server breaks down, the queue system stops sending data for a random time denoted as h_n with a general distribution; subsequently, the server goes on from the point at which it broke down. We assume that based on long-term statistics of the stable original TSN stream queue system, the mean time $E[h)_n]$ and the second moment $E[h_n^2]$ are known. When the task offloading streams arrive at $Q_{p,n}$, as the GCL and higher priority queues are unchanged and only the customer arrival rate increases in $Q_{p,n}$, we suppose that the server of queue $Q_{p,n}$ still breaks down at the exponential rate α_n, and the pause time still follows the previous general distribution.

Here, we discuss the average time a customer spends in queue $Q_{p,n}$ with server breaking down. Let $\mathscr{H}$ denote the random variable "cost time" that includes service and pause time of server breaking down, k denotes the number of times the server breaks down, and $h_n^1, h_n^2, \ldots, h_n^k$ denote the amount of time the server waits for in each breakdown period. Then, it can be found that

$$\mathscr{H} = \sum_{i=1}^{k} h_n^i + \Omega_{*,n}. \tag{8}$$

According to the Pollaczek–Khintchine formula [31], the average waiting time before a customer begins to be served is calculated as follows:

$$\overline{\mathscr{W}}_{*,n} = \frac{\lambda_p E\left[\mathscr{H}^2\right]}{2\left(1-\lambda_p E[\mathscr{H}]\right)}. \tag{9}$$

To obtain $E[\mathscr{H}]$ and $E[\mathscr{H}^2]$, we assume that a customer in $Q_{p,n}$ requires service time s'; thus, we can obtain the following expressions:

$$E\left[\mathscr{H}|\Omega_{*,n}=s'\right] = E\left[\sum_{i=1}^{k} h_n^i|\Omega_{*,n}=s'\right] + s', \tag{10}$$

$$\mathrm{Var}\left(\mathscr{H}|\Omega_{*,n}=s'\right) = \mathrm{Var}\left(\sum_{i=1}^{k} h_n^i|\Omega_{*,n}=s'\right). \tag{11}$$

As the number of times the server was interrupted while a customer is in service is a Poisson random variable with mean $\alpha_n s'$, the sum $\sum_{i=1}^{k} h_n^i$ is also a compound Poisson variable with mean $\alpha_n s'$, when $\Omega_{*,n} = s'$. Consequently, it can be seen that

$$E\left[\sum_{i=1}^{k} h_n^i|\Omega_{*,n}=s'\right] = \alpha_n s' E\left[h_n\right], \tag{12}$$

$$\mathrm{Var}\left(\sum_{i=1}^{k} h_n^i|\Omega_{*,n}=s'\right) = \alpha_n s' E\left[h_n^2\right]. \tag{13}$$

Using equations (9)–(13), it can be found that

$$E\left[\mathscr{H}|\Omega_{*,n}\right] = \alpha_n \Omega_{*,n} E\left[h_n\right] + \Omega_{*,n}, \tag{14}$$

$$\mathrm{Var}\left(\mathscr{H}|\Omega_{*,n}\right) = \alpha_n \Omega_{*,n} E\left[h_n^2\right]. \tag{15}$$

Furthermore, we see that

$$\begin{aligned} E[\mathscr{H}] &= E\left[E\left[\mathscr{H}|\Omega_{*,n}\right]\right] \\ &= E\left[\Omega_{*,n}\right]\left(1+\alpha_n E\left[h_n\right]\right), \end{aligned} \tag{16}$$

$$\begin{aligned} \mathrm{Var}(\mathscr{H}) &= E\left[\mathrm{Var}\left(\mathscr{H}|\Omega_{*,n}\right)\right] + \mathrm{Var}\left(E\left[\mathscr{H}|\Omega_{*,n}\right]\right) \\ &= \alpha_n E\left[\Omega_{*,n}\right] E\left[h_n^2\right] + \left(1+\alpha_n E\left[h_n\right]\right)^2 \mathrm{Var}\left(\Omega_{*,n}\right). \end{aligned} \tag{17}$$

The second moment can then be obtained as follows:

$$\begin{aligned} E\left[\mathscr{H}^2\right] &= \mathrm{Var}(H) + (E[\mathscr{H}])^2 \\ &= \alpha_n E\left[h_n^2\right] E\left[\Omega_{*,n}\right] + E\left[\Omega_{*,n}^2\right]\left(1+\alpha_n E\left[h_n\right]\right)^2. \end{aligned} \tag{18}$$

Finally, substituting equations (16) and (18) into equation (9) and assuming that the inequality $1 > (\tilde{\lambda}_{p,n} + \sum_{i=1}^{M} \lambda_{i,n}) E[\Omega_{*,n}](1+\alpha_n E[h_n])$ is satisfied, the average waiting time can be written as follows:

$$\begin{aligned} &\overline{\mathscr{W}}_{*,n} = \frac{\left(\tilde{\lambda}_{p,n} + \sum_{i=1}^{M} \lambda_{i,n}\right)\left(\alpha_n E\left[h_n^2\right] E\left[\Omega_{*,n}\right] + E\left[\Omega_{*,n}^2\right]\left(1+\alpha_n E\left[h_n\right]\right)^2\right)}{2\left(1-\left(\tilde{\lambda}_{p,n} + \sum_{i=1}^{M} \lambda_{i,n}\right) E\left[\Omega_{*,n}\right]\left(1+\alpha_n E\left[h_n\right]\right)\right)}, \\ &\text{s.t. } 1 > \left(\tilde{\lambda}_{p,n} + \sum_{i=1}^{M} \lambda_{i,n}\right) E\left[\Omega_{*,n}\right]\left(1+\alpha_n E\left[h_n\right]\right). \end{aligned} \tag{19}$$

Thus, the average time that the members of the hybrid stream spend in queue $Q_{p,n}$ is calculated as follows:

$$\overline{Q}_{*,n} = \overline{\mathscr{W}}_{*,n} + E(\mathscr{H}) = \overline{\mathscr{W}}_{*,n} + E\left[\Omega_{*,n}\right]\left(1+\alpha_n E\left[h_n\right]\right). \tag{20}$$

Suppose that when task offloading streams reach the E_n, E_n immediately provides parallel computing services, which offer CPU resources with frequency f_n for each offloaded task. To guarantee that the offloading computing service requirement is beyond the maximum capacity of E_n, the parameter Λ_n is used to denote the available maximum offloading task arrival rate from all edge devices to E_n, i.e., $\sum_{i=1}^{M} \lambda_{i,n} \le \Lambda_n$. In E_n, the average computation time for offloaded tasks from D_i is $\overline{C}_{i,n} = E[\ell_{i,*}] \times \rho_i / f_n$. Considering that the data transmission speed on the TSN wire is fast (e.g., 1000 Mbps) and the offloaded task computation result's volume is typically small (maybe a few bytes), the time of computation result delivering

back can be neglected. Moreover, the time for the task offloading stream in the switch is mostly spent in $Q_{p,n}$, then the average remote computing time for all tasks offloaded from edge device D_i to E_n can be written as follows:

$$\overline{T}_{i,n} = \overline{Q}_{*,n} + \overline{C}_{i,n}. \tag{21}$$

Finally, the average response time for all the tasks generated in edge device D_i can be obtained as follows:

$$\overline{T}_i = \frac{\lambda_{i,\Delta} + \lambda_{i,0}}{\lambda_i}\overline{T}_{i,0} + \sum_{j=1}^{N}\frac{\lambda_{i,j}}{\lambda_i}\overline{T}_{i,j}. \tag{22}$$

4. Problem Formulation and Algorithm

4.1. Game Formulation. In the TSN computation offloading scenario, each edge device is independent and selfish with the aim of utilizing the maximum ECS computing resources possible to reduce its own response time; this represents a suitable situation for game theory to be used to model the competition process among all edge devices. Considering the edge device set $\mathscr{M} = \{D_1, D_2, \ldots, D_M\}$ as the game players, the arrival rate allocation policy set $\mathscr{S}_i = (\lambda_{i,0}, \lambda_{i,1}, \ldots, \lambda_{i,N}) \in R^{N+1}$ as the strategies of player D_i (where $\lambda_i = \lambda_{i,\Delta} + \lambda_{i,0} + \lambda_{i,1} + \ldots + \lambda_{i,N}$), and the Cartesian products of all individual strategy set, i.e., $\mathbb{S} = \mathscr{S}_1 \times \mathscr{S}_2 \times \ldots \times \mathscr{S}_M$, as the strategy space, we give the multitask offloading game as follows:

Definition 1. Multitask offloading game: the multitask offloading game, $\mathscr{G}$, which is composed of the player set $\mathscr{M}$, the offloading strategy space $\mathbb{S}$, and the utility function set $\{U_i\}, \forall i \in \mathscr{M}$, is represented as follows:

$$\mathscr{G} = [\mathscr{M}, \mathbb{S}, \{U_i\}_{i\in\mathscr{M}}]. \tag{23}$$

We expect that within limited rounds of decision-making, the players can reach a certain consensus, i.e., each player benefits most in that state. Thus, we now introduce Nash Equilibrium as follows:

Definition 2. Nash equilibrium [33]: the strategy profile $\{\mathscr{S}_i^*\}_{i\in\mathscr{M}}$ is a Nash equilibrium of multitask offloading game $\mathscr{G}$ if and only if

$$U_i(\mathscr{S}_i^*, \mathscr{S}_{-i}) \le U_i(\mathscr{S}_i', \mathscr{S}_{-i}), \forall \mathscr{S}_i' \in \mathscr{S}_i, \tag{24}$$

where $\mathscr{S}_{-i}$ is the offloading strategies of computing devices except D_i. That means there is no more optimal strategy than $\mathscr{S}_i^*$ to get more benefit based on the utility function. The challenge in fully defining the game formulation (equation (23)) is to find an appropriate utility function, which allows all players to maximize the benefits from their own perspectives and achieve Nash equilibrium. Here, we adopt the exact potential game theory as defined below to design the utility function.

Definition 3. Exact potential game [34]: the multitask offloading game, $\mathscr{G}$, is an exact potential game if and only if there exists a potential function $F(\mathscr{S})$: $\mathbb{S} \longrightarrow$ R that satisfies

$$\begin{aligned} U_i(s_i, s_{-i}) - U_i(s_i', s_{-i}) = F(s_i, s_{-i}) - F(s_i', s_{-i}), \\ \forall s_i, s_i' \in \mathscr{S}_i; \forall s_{-i} \in \mathscr{S}_{-i}; \forall i \in \mathscr{M}. \end{aligned} \tag{25}$$

As the existence of Nash equilibrium is a fundamental property of exact potential games, we construct a potential function, F and a utility function, U_i, to ensure that the multitask offloading game, $\mathscr{G}$, is an exact potential game using the backward method [34]. We let $F(\mathscr{S})$ denote the whole system benefits that can be achieved via task offloading computing rather than the situation in which all tasks are locally executed. $F(\mathscr{S})$ can be represented in computation time, as the greater benefit is equated with a lesser time cost. Thus, we see that.

$$F(\mathscr{S}) = \sum_{k=1}^{M}\overline{T}_{k,0}^{l} - \sum_{k=1}^{M}\frac{(\lambda_{k,\Delta} + \lambda_{k,0})\overline{T}_{k,0}}{\lambda_k} - \sum_{k=1}^{M}\sum_{j=1}^{N}\frac{\lambda_{k,j}\overline{T}_{k,j}}{\lambda_k}. \tag{26}$$

Subsequently, considering equations (1)–(22), a decomposition can be obtained as follows:

$$\begin{aligned} \mathrm{F}(\mathscr{S}_i, \mathscr{S}_{-i}) &= \sum_{k=1}^{M}\overline{T}_{k,0}^{l} - \frac{(\lambda_{i,\Delta} + \lambda_{i,0})\overline{T}_{i,0}}{\lambda_i} - \sum_{k=1,k\neq i}^{M}\frac{(\lambda_{k,\Delta} + \lambda_{k,0})\overline{T}_{k,0}}{\lambda_k} \\ &\quad - \sum_{k=1}^{M}\sum_{j=1}^{N}\frac{\lambda_{k,j}\overline{Q}_{*,j}}{\lambda_k} - \sum_{j=1}^{N}\frac{\lambda_{k,j}\overline{Q}_{*,j}}{\lambda_k} - \sum_{k=1,k\neq i}^{M}\sum_{j=1}^{N}\frac{\lambda_{k,j}\overline{C}_{i,j}}{\lambda_k} \\ &= -\frac{\lambda_{i,\Delta} + \lambda_{i,0}}{\lambda_i}\bullet\frac{\rho_i^2\left(\lambda_{i,\Delta}E\left[\ell_{i,\Delta}^2\right] + \lambda_{i,0}E\left[\ell_{i,*}^2\right]\right)}{2\widehat{f}_i^2 - 2\widehat{f}_i\rho_i\left(\lambda_{i,\Delta}E\left[\ell_{i,\Delta}\right] + \lambda_{i,0}E\left[\ell_{i,*}\right]\right)} \\ &\quad - \frac{\rho_i\left(\lambda_{i,\Delta}E\left[\ell_{i,\Delta}\right] + \lambda_{i,0}E\left[\ell_{i,*}\right]\right)}{\lambda_i\widehat{f}_i} - \sum_{k=1}^{M}\sum_{j=1}^{N}\frac{\lambda_{k,j}}{\lambda_k}\left\{\overline{\mathscr{W}}_{*,j} + E\left[\Omega_{*,j}\right]\left(1 + \alpha_j E\left[h_j\right]\right)\right\} - \sum_{j=1}^{N}\frac{\lambda_{i,j}E\left[\ell_{i,*}\right]\rho_i}{\lambda_i f_j} + \\ &\quad \underbrace{\sum_{k=1}^{M}\overline{T}_{k,0}^{l} - \sum_{k=1,k\neq i}^{M}\sum_{j=1}^{N}\frac{\lambda_{k,j}E\left(\ell_{k,*}\right)\rho_k}{\lambda_k f_j} - \sum_{k=1,k\neq i}^{M}\frac{(\lambda_{k,\Delta} + \lambda_{k,0})\overline{T}_{k,0}}{\lambda_k}}_{\text{Non-contributing term } \phi_i(S_{-i})}. \end{aligned} \tag{27}$$

Then, we can obtain the utility function of the potential game with $U_i(\mathcal{S}_i, \mathcal{S}_{-i}) = F(\mathcal{S}_i, \mathcal{S}_{-i}) - \phi_i(\mathcal{S}_{-i})$, where $\phi_i(S_{-i})$ is a noncontributing term from the perspective of player D_i. Thus, the utility function of the exact potential game $\mathcal{G}$ is given as follows:

$$U_i(\mathcal{S}_i, \mathcal{S}_{-i}) = -\frac{\lambda_{i,\Delta} + \lambda_{i,0}}{\lambda_i} \bullet \frac{\rho_i^2\left(\lambda_{i,\Delta}E\left[\ell_{i,\Delta}^2\right] + \lambda_{i,0}E\left[\ell_{i,*}^2\right]\right)}{2\widehat{f}_i^2 - 2\widehat{f}_i\rho_i\left(\lambda_{i,\Delta}E\left[\ell_{i,\Delta}\right] + \lambda_{i,0}E\left[\ell_{i,*}\right]\right)} - \frac{\rho_i\left(\lambda_{i,\Delta}E\left[\ell_{i,\Delta}\right] + \lambda_{i,0}E\left[\ell_{i,*}\right]\right)}{\lambda_i\widehat{f}_i}$$
$$- \sum_{k=1}^{M}\sum_{j=1}^{N}\frac{\lambda_{k,j}}{\lambda_k}\left\{\overline{\mathscr{W}}_{*,j} + E\left[\Omega_{*,j}\right]\left(1 + \alpha_j E\left[h_j\right]\right)\right\} - \sum_{j=1}^{N}\frac{\lambda_{i,j}E\left[\ell_{i,*}\right]\rho_i}{\lambda_i f_j}. \tag{28}$$

4.2. The Algorithm. Based on the utility function U_i, the best response strategy of each edge device is described.

Definition 4. Best response strategy: let $\mathcal{S}_{-i}$ represent the strategies of all the edge devices except device D_i, the best response strategy of D_i is given by

$$\mathcal{S}_i^* = \underset{\mathcal{S}_i}{\operatorname{argmax}}\, U_i(\mathcal{S}_i, \mathcal{S}_{-i}), \tag{29a}$$

$$\lambda_{i,\Delta} + \sum_{j=0}^{N}\lambda_{i,j} = \lambda_i, \tag{29b}$$

$$\sum_{k=1}^{M}\lambda_{k,j} \le \Lambda_j, \tag{29c}$$

$$1 > \left(\lambda_{i,\Delta} + \lambda_{i,0}\right)E\left[C_{i,0}\right], \tag{29d}$$

$$1 > \left(\widetilde{\lambda}_{p,n} + \sum_{i=1}^{M}\lambda_{i,n}\right)E\left[\Omega_{*,n}\right]\left(1 + \alpha_n E\left[h_n\right]\right). \tag{29e}$$

Equation (29a) represents the choice of D_i to pursue minimal time cost. The constraint equation (29b) guarantees that the decision of the task arrival rate allocation meets the total arrival rate requirement for each edge device. The constraint equation (29c) ensures that the workload of ECSs providing remote computing services is not beyond the upper limit of its capacity. Equations (29d) and (29e) guarantee that the queuing system is in a serviceable state.

According to the aforementioned system model and game formulation, the distributed and sequential decision-making algorithm for multi-task offloading (DSDA-MO) was designed. A summary of this algorithm is given in Algorithm 1. After the strategies of each edge device are initialized, a token is passed among edge devices, and the device in possession of the token calculates its best response strategy based on all the decisions of the other devices. The algorithm runs (line 3–8) until the $\mathrm{MSE}(\mathcal{S}^{*(R)}, \mathcal{S}^{*(R-1)})$, the mean square error between the strategy of the current round and the strategy of the previous round, is less than accuracy parameter ξ (e.g., $\xi = 10^{-10}$). $\mathrm{MSE}(\mathcal{S}^{*(R)}, \mathcal{S}^{*(R-1)})$ is defined as follows:

$$\mathrm{MSE}\left(\mathcal{S}^{*(R)}, \mathcal{S}^{*(R-1)}\right) = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N}\left(\lambda_{i,j}^{*(R_i)} - \lambda_{i,j}^{*(R_i-1)}\right)^2}{M}. \tag{30}$$

In each round of Algorithm 1, the equations (29a)–(29e). Problem could been solved via the Lagrange multiplier method with the algorithm of iterative search in feasible region, and the complexity is $O(MN(\log 1/\delta)^2)$, where δ is the search interval. Thus, the complexity of Algorithm 1 is $O(RMN(\log 1/\delta)^2)$, where R is the number of rounds mainly determined by the accuracy parameter ξ in line 9.

5. Numerical Results

In this section, the results of numerical experiments are reported to evaluate the proposed system model and the DSDA-MO algorithm. In the simulation setup, we consider M edge devices ($D_1, D_2, \ldots, D_i$, $i = 1, 2, \ldots, M$) and N ECSs ($E_1, E_2, \ldots, E_j$, $j = 1,2, \ldots, N$) with the following parameters: $\lambda_i = 3.4 + 0.1(i-1)$, $\lambda_{i,\Delta} = 0.5 + 0.05(i-1)$, $E[\ell_{i,\Delta}] = 5 + (i-1)M$bit, $E[\ell_{i,\Delta}^2] = 1.5(E[\ell_{i,\Delta}])^2$, $E[\ell_{i,*}] = 100 + (i-1)M$bit, $E[\ell_{i,*}^2] = 1.6(E[\ell_{i,*}])^2$, $\rho_i = 1 + 0.1(i-1)$BI/Mbit, $E[g_{p,j}] = 40 + 0.5(j-1)M$bit, $E[g_{p,j}^2] = 1.3(E[g_{p,j}])^2$, $E[h_j] = 0.01 + 0.001(j-1)$s, $E[h_j^2] = 1.1E[h_j]^2$, $\alpha_j = 0.03 + 0.001(j-1)$, $\widetilde{\lambda}_{p,j} = 1.2 + 0.1(j-1)$, $\Lambda_j = 7 + 0.5(j-1)$, $\widehat{f}_i = 0.8 + 0.1(i-1)$ GHz, $f_j = 3.2 + 0.2(j-1)$ GHz, and $v = 1000$ Mbps, for all $1 \le i \le M$, $1 \le j \le N$. The simulation is constructed on MATLAB platform.

5.1. Nash Equilibrium of the Game. First, we set $M = 6$, $N = 5$ to evaluate the results of Nash equilibrium, i.e., the final optimal strategy $\mathcal{S}_i^* = (\lambda_{i,0}^*, \lambda_{i,1}^*, \ldots, \lambda_{i,N}^*)$ for $1 \le i \le M$. Figure 2 shows the curves describing the utility of the edge devices ($D_1, \ldots, D_6$) found at each round of the DSDA-MO algorithm. It seems that the utilities fluctuate wildly in the initial rounds of calculations and then enter a stable zone with low variation. Figure 2 shows the average time cost at each round, and it illustrates that the average time cost of each device remains near a stable value after about 200 rounds. The results in Figures 2 and 3 verify that the game reaches the approximate Nash equilibrium after finite rounds of calculation in the proposed algorithm.

To further evaluate the speed of convergence of the Nash equilibrium, we list the strategies, i.e.,

$(\lambda_{5,0}^{*(R)}, \lambda_{5,1}^{*(R)}, \ldots, \lambda_{5,5}^{*(R)})$ and $(\lambda_{6,0}^{*(R)}, \lambda_{6,1}^{*(R)}, \ldots, \lambda_{6,5}^{*(R)})$, from 30 to 90 rounds in Tables 3 and 4, and give the curves of strategies at each round (as shown in Figure 4) by using the edge device D_5 and D_6 as examples. The results in Tables 3 and 4 and Figure 4 show that $(\lambda_{i,1}^*, \ldots, \lambda_{i,N}^*)$ changes significantly in each round at the early stage until round number 200; from this point on, the optimal strategy remains in a stable state, which is consistent with the findings presented in Figures 2 and 3. As discussed earlier in Algorithm 1, the round number to get approximate Nash equilibrium mainly depends on the accuracy parameter ξ. Table 5 presents the relationship between the round number and the value of ξ, listed as $10^{-1}, 10^{-2}, \ldots, 10^{-10}$. It shows that the round number increases with the improvement of the accuracy requirement. That is to say, a smaller value of ξ leads to a greater round number. We can select the appropriate parameters according to the application demand with the appropriate balance between the Nash equilibrium accuracy and the algorithm running time.

5.2. Benefit of the Game. In this subsection, we focus on analyzing the benefit that edge devices obtain from the proposed game. Let accuracy parameter $\xi = 10^{-8}$; after 220 rounds of calculation in Algorithm 1, we obtain the final optimal strategy profile $\mathcal{S}^*$. Figure 5 presents the arrival rate of the un-offloadable tasks (known parameter), the sum of the optimal arrival rates of the tasks offloaded from D_m to all ECS*s* (i.e., $\sum_{j=1}^{N} \lambda_{m,j}^*$, $1 \le m \le M$), and the optimal arrival rate of the offloadable local computing tasks (i.e., $\lambda_{m,0}^*$) in edge devices. It is found that $\sum_{j=1}^{N} \lambda_{m,j}^*$ increases as the arrival rate of all tasks generated on each edge device increases with $\lambda_i = 3.4 + 0.1(i-1)$ in our experiment settings. In other words, the amount of the offloaded tasks increases as the computation workload ascends in edge devices. Figure 6 illustrates the average time with the decision of all locally-computed tasks versus the average time with optimal strategy profile $\mathcal{S}^*$ obtained by Algorithm 1. Compared with the method of all tasks computing locally, our proposed algorithm brings significant time-saving advantages for each device, where D_6 obtains 13.03% decrement in time cost.

To study the profitability of the proposed approach, we define the task offloading proportion $\mathcal{R}(i,n) \in [0,1]$, as

$$\mathcal{R}(i,n) = \frac{\sum_{j=1}^{n} \lambda_{i,j}^*}{\sum_{j=0}^{n} \lambda_{i,j}^*}, \tag{31}$$

where $\sum_{j=1}^{N} \lambda_{i,j}^*$ is the sum of the arrival rates of the tasks offloaded to $E_1, E_2, \ldots, E_N$ from D_i, and $\sum_{j=0}^{N} \lambda_{i,j}^*$ is the sum of the arrival rates of all offloadable tasks on D_i. The larger $\mathcal{R}(i,n)$ means more benefits obtained from task offloading for D_i. Figure 7 illustrates that the task offloading proportions of $D_1, D_2, \ldots, D_6$ increase in order as their computation workload increases accordingly, and the task offloading proportion increment of D_6 reaches more than 25%.

We also study the workload of the edge computing servers in the game. Figure 8 illustrates the final amount of the workload in the target ECS set $\mathcal{N}$, i.e., $\{E_1, E_2, ..., E_N\}$, and the final decision that how many tasks on each edge device will choose each ECS. The decision is indicated by the arrival rate $\widehat{\lambda}_j$ from all edge devices to E_j, $1 \le j \le N$. From Figure 8, we can see that although the processing frequencies of the members of set $\mathcal{N}$ increase successively in our experiment settings, the sum of the task arrival rates from the edge devices to ECS*s*, i.e., $\widehat{\lambda}_1, \widehat{\lambda}_2, \ldots, \widehat{\lambda}_N$, descends gradually. In other words, ECS*s* gradually lost their attraction to the tasks offloaded from edge devices as the processing frequency increases successively. This is because each offloaded task needs to consider both the computation time and the transmission time and makes a decision based on the sum of them. Although high processing frequency of the target ECS leads to low computation time for the offloaded task, the sum of the task arrival rates may also be small due to a possible high transmission time cost. In our experimental settings, it is assumed that the busy degree of the TSN links to $E_1, E_2, \ldots, E_N$ increase successively. That is to say, the average interrupted time of the TSN output queues corresponding to each target ECS, i.e., $E[h_n]$, increases orderly. This experiment setting results in an increase in the transmission time of the same task to $E_1, E_2, \ldots, E_N$ orderly. Hence, the number of the tasks selected to be offloaded to ECS*s*, namely, the task arrival rates to ECS*s*, gradually decreases in Figure 8. Especially, almost none of the edge devices here decided to offload their task to E_5.

To analyze the influence of the number of the edge devices, M, and the number of ECS*s*, N, on the task offloading decision-making result, we construct an experiment with $M = \{10, 15, 20, 25, 30\}$, $N = \{20, 25, 30\}$, and $\xi = 10^{-5}$. Figure 9 shows the average response time of the tasks in the edge devices with different M and N, which reflects the impact on the performance of the entire edge computing system. Obviously, the average response time of the edge computing system increases as M increases when N is fixed. Inversely, the average response time decrease as N increases when M is fixed.

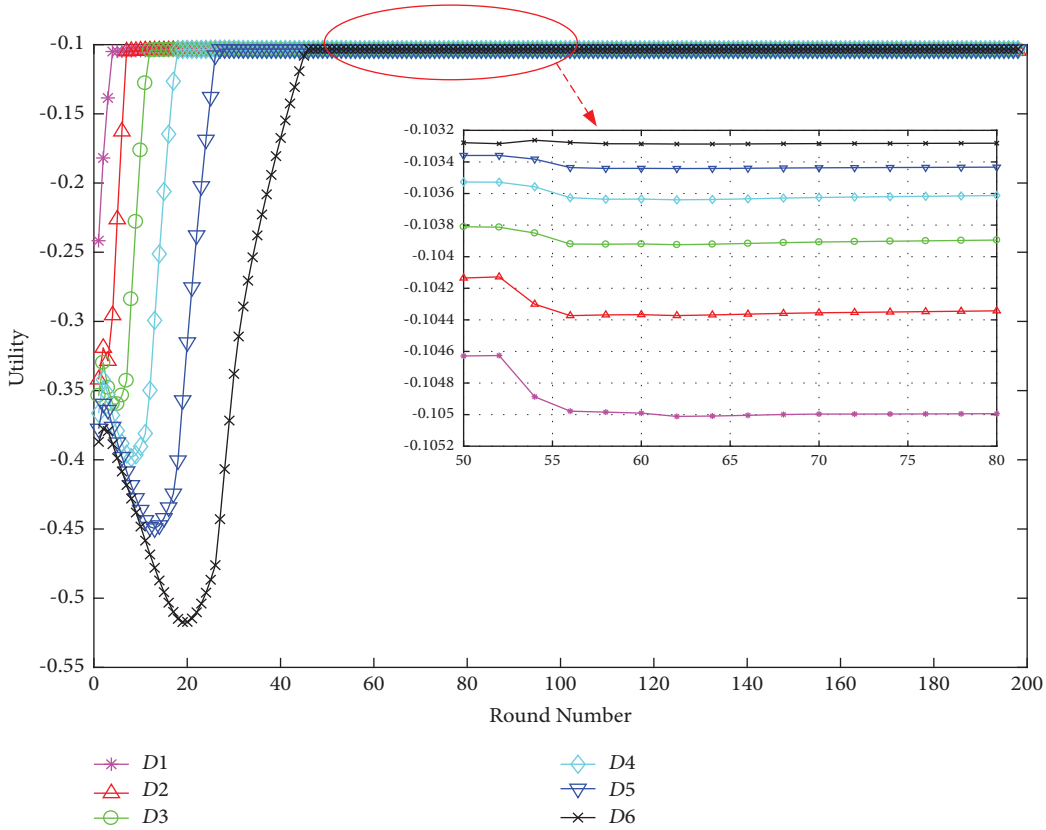

Figure 2: Utility of the edge devices obtained using the proposed algorithm at each round. The curves of utility from round 50 to 80 are zoomed.

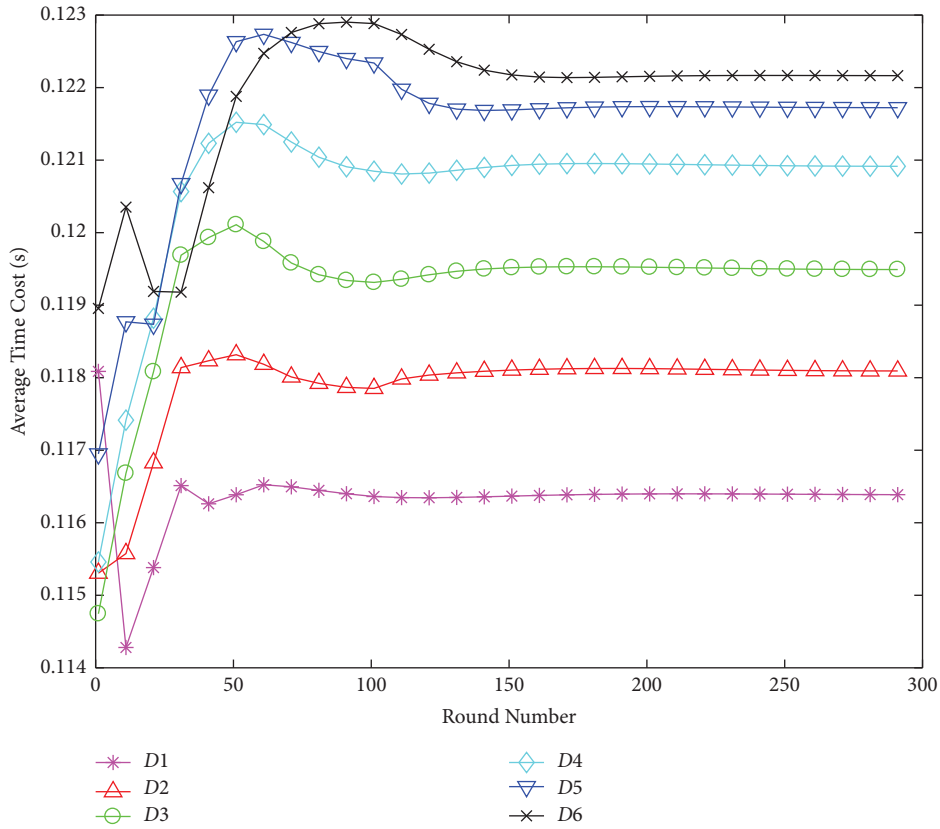

Figure 3: Average time cost of the edge devices at each round.

Input: λ_i, $\lambda_{i,\Delta}$, ρ_i, $E[\ell_{i,\Delta}]$, $E[\ell_{i,\Delta}^2]$, $E[\ell_{i,*}]$, $E[\ell_{i,*}^2]$, $E[h_j]$, $E[h_j^2]$, $\widetilde{\lambda}_{p,j}$, f_i, $E[\mathcal{Q}_{p,j}]$, $E[\mathcal{Q}_{p,j}^2]$, α_j, $\widetilde{\lambda}_{p,j}$, Λ_j, ν and $\widehat{f}_j$, for all $1 \le i \le M$ and $1 \le j \le N$

(1) **Initialization:** $\mathcal{S}_i^{*(0)} = (\lambda_{i,0}^{*(0)}, \lambda_{i,1}^{*(0)}, \ldots, \lambda_{i,N}^{*(0)})$ and $R_i \leftarrow 0$ for all $1 \le i \le M$, and give the token to first edge device D_1.
(2) **Repeat:**
(3) Player D_i waits for the token
(4) Player D_i collects and updates all the other edge devices' decisions, $\mathcal{S}_{-i}^*$
(5) $R_i \leftarrow R_i + 1$
(6) Player D_i calculates the best response strategy
$\mathcal{S}_i^{*(R_i)} \leftarrow \operatorname{argmax} U_i(\mathcal{S}_i, \mathcal{S}_{-i})$,
Subject to (29b)–(29e)
(7) Player D_i broadcasts $\mathcal{S}_i^*$ to all the other edge devices
(8) Player D_i sends the token to the next node
(9) **Until** $\mathrm{MSE}(\mathcal{S}^{*(R)}, \mathcal{S}^{*(R-1)}) < \xi$
(10) **Output:** $\mathcal{S}^* \leftarrow (\mathcal{S}_1^{*(R_1)}, \mathcal{S}_2^{*(R_2)}, \ldots, \mathcal{S}_N^{*(R_N)})$

Algorithm 1: Distributed and sequential decision-making algorithm for multitask offloading (DSDA-MO).

Table 3: Convergence of the Nash equilibrium in example D_5.

Round	$\lambda_{5,0}^*$	$\lambda_{5,1}^*$	$\lambda_{5,2}^*$	$\lambda_{5,3}^*$	$\lambda_{5,4}^*$	$\lambda_{5,5}^*$
30	2.9813121	0.1348270	0.1525576	0.1665572	0.1778081	0.1869380
40	2.9681589	0.1289364	0.1517291	0.1698735	0.1846137	0.1966883
50	2.9622163	0.1212920	0.1493441	0.1717868	0.1901334	0.2052274
60	2.9623449	0.1122118	0.1456737	0.1725337	0.1945440	0.2126918
70	2.9628887	0.1029620	0.1419134	0.1732544	0.1989031	0.2200784
80	2.9641592	0.0936193	0.1380922	0.1739339	0.2031423	0.2270532
90	2.9642554	0.0847129	0.1346802	0.1749801	0.2076104	0.2337610

Table 4: Convergence of the Nash equilibrium in example D_6.

Round	$\lambda_{6,0}^*$	$\lambda_{6,1}^*$	$\lambda_{6,2}^*$	$\lambda_{6,3}^*$	$\lambda_{6,4}^*$	$\lambda_{6,5}^*$
30	2.9683538	0.165065	0.178211	0.188503	0.196666	0.2032013
40	2.9652401	0.157901	0.175636	0.189742	0.201114	0.2103667
50	2.9598825	0.150956	0.173430	0.191459	0.206121	0.2181514
60	2.9591486	0.142797	0.170189	0.192285	0.210342	0.2252379
70	2.9594829	0.134183	0.166648	0.192932	0.214463	0.2322909
80	2.9605833	0.125198	0.162884	0.193462	0.218540	0.2393330
90	2.9608394	0.116203	0.159237	0.194203	0.222882	0.2466355

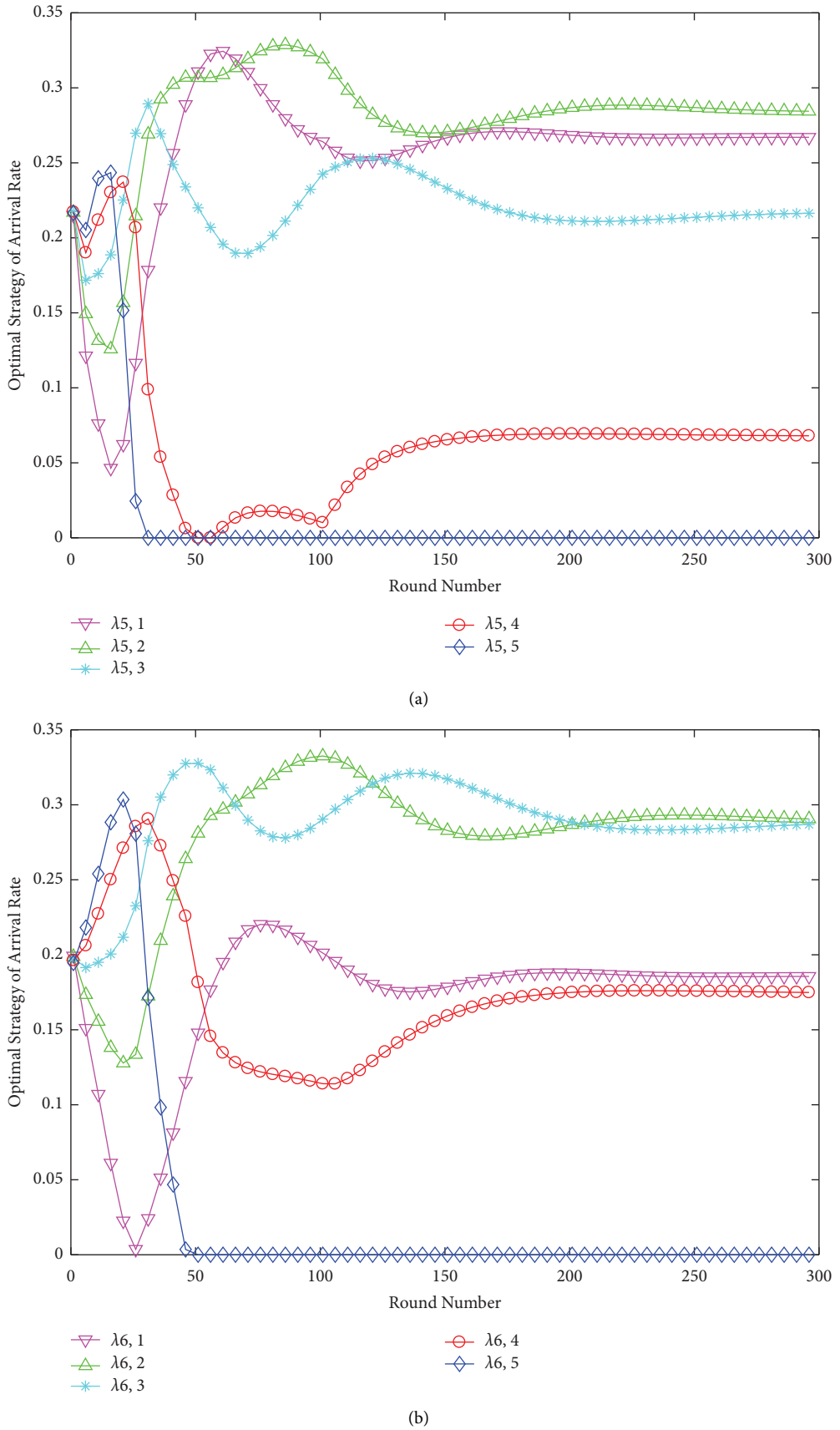

Figure 4: Optimal strategy of the arrival rate obtained by using the proposed algorithm at each round: edge device D_5 and D_6 as examples. (a) Edge device D_5 and (b) edge device D_6.

Table 5: Round number with the value of accuracy parameter ξ.

Accuracy parameter ξ	Round number
10^{-1}	3
10^{-2}	12
10^{-3}	21
10^{-4}	35
10^{-5}	64
10^{-6}	127
10^{-7}	177
10^{-8}	220
10^{-9}	289
10^{-10}	327

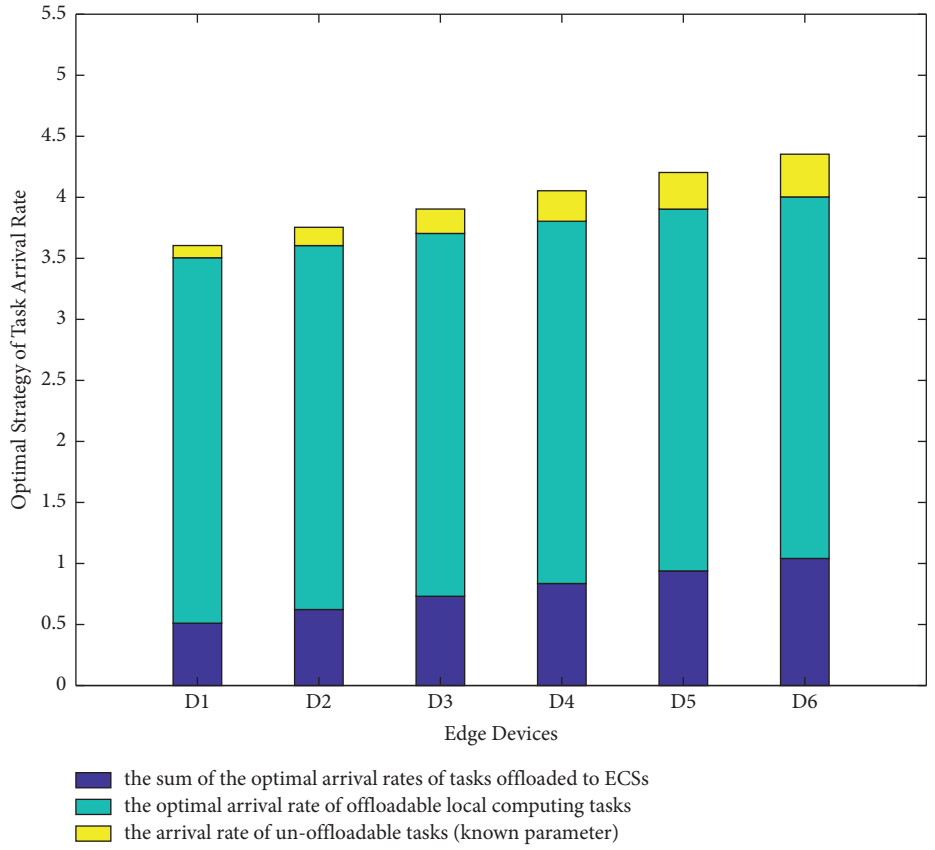

Figure 5: Optimal strategies of the task arrival rate consisting of the arrival rate of the unoffloadable tasks (known parameter), the arrival rate of the offloadable local computing tasks, and the arrival rate of the tasks offloaded to ECSs.

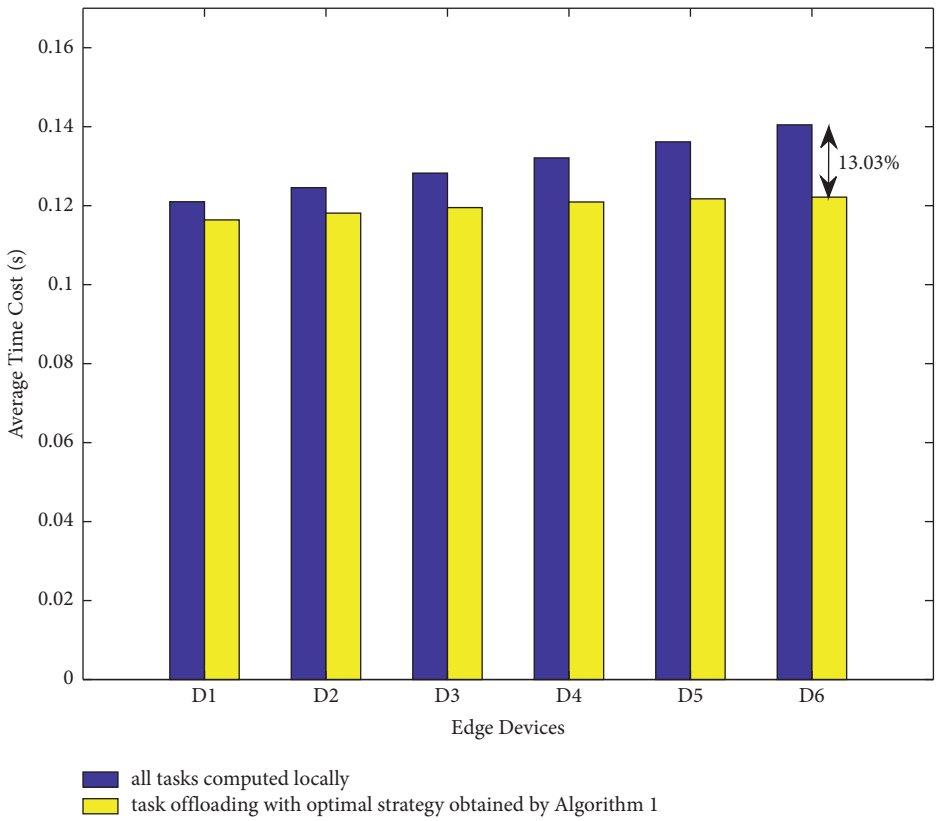

Figure 6: Average time cost with the decision of all tasks computed locally vs. average time cost with optimal strategy profile $\mathcal{S}^*$ obtained by algorithm 1.

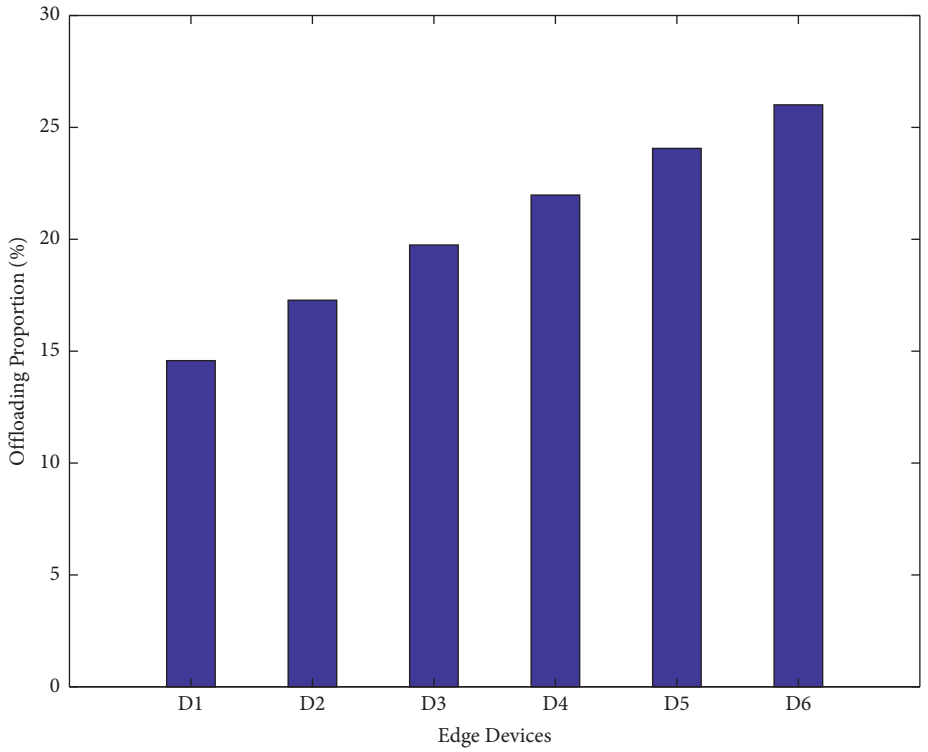

Figure 7: Task offloading proportion of each edge device.

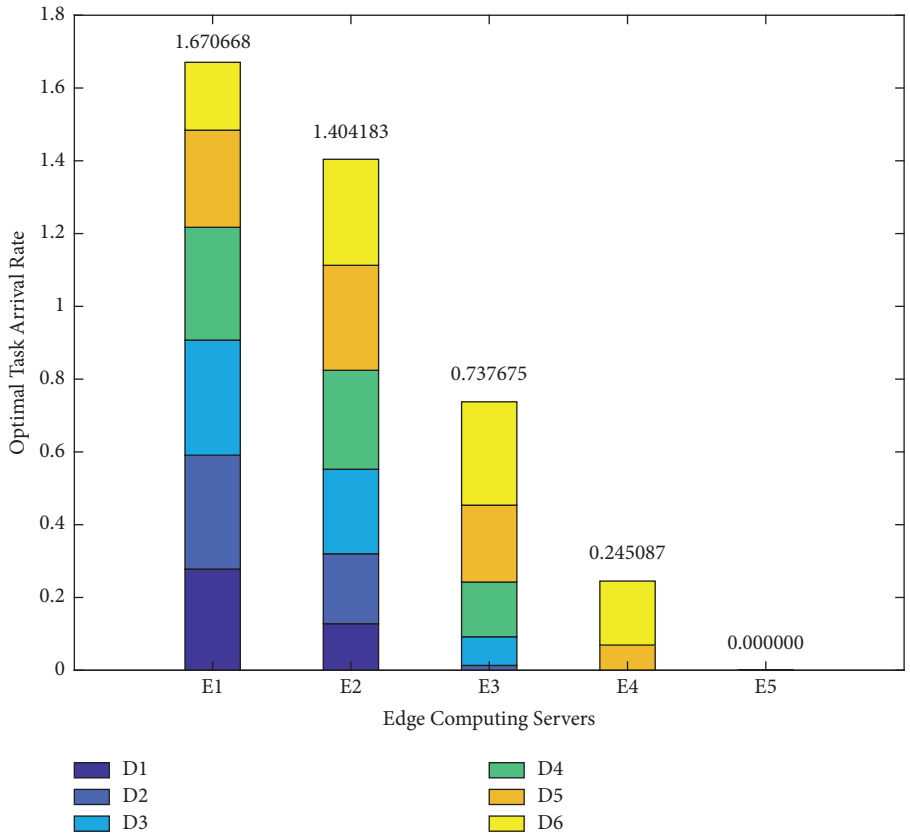

Figure 8: Optimal arrival rates of the tasks to each ECS.

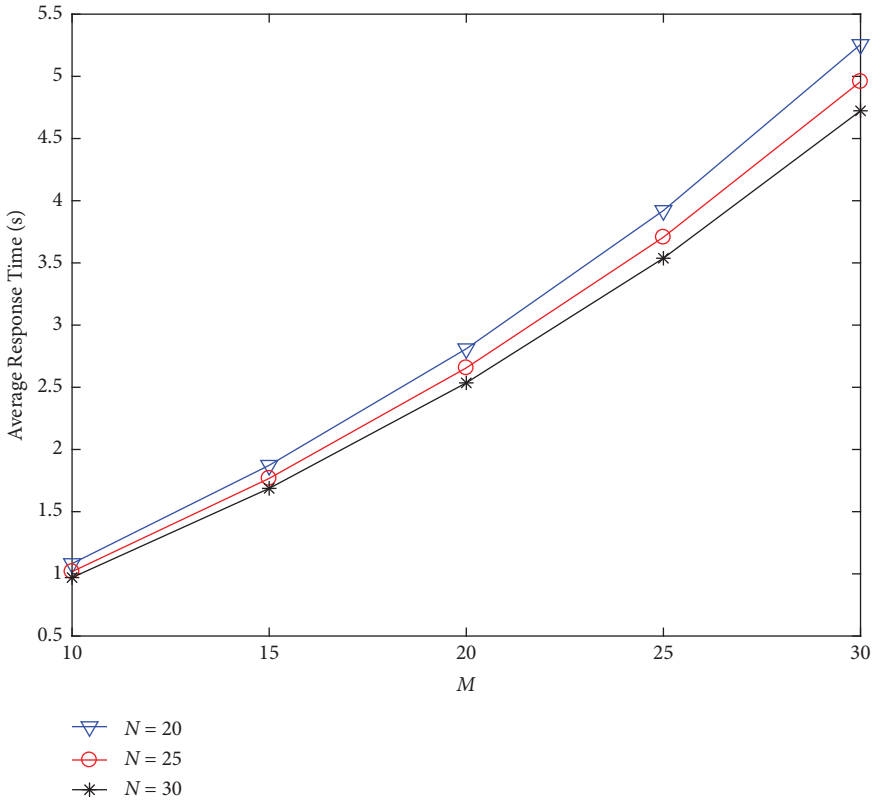

Figure 9: The average response time of the tasks in all edge devices with different M and N values.

6. Conclusion

In this paper, we adopt the queueing theory to establish a time-cost model for task offloading stream transmission to resolve the problem of task offloading decision-making for edge computing in TSN. Meanwhile, we construct a utility function with the backward method to formulate the task offloading competition among edge devices as a game model. To find the Nash equilibrium of the proposed game, we design a distributed and sequential decision-making algorithm for multitask offloading. Finally, the exist of Nash equilibrium, the speed of convergence of Nash equilibrium, and the relationship between the number of calculation rounds and accuracy parameter ξ are investigated through numerical experiments. Furthermore, to study the benefit of the proposed game theoretic approach, we analyze and compare the optimal strategy of task arrival rate, average time cost, and task offloading proportion among each edge device. The experiment results demonstrate that the TSN edge devices can obtain the optimal multi-task offloading strategies within finite rounds of calculation, which significantly reduce the average time cost of task computation, by utilizing the proposed model and approach.

7. Future Work

For future work, we will study the scenario where the tasks are generated in the queues of edge devices and are processed in the queues of ECS*s*, involving a multilevel queuing system. The goal is to develop efficient algorithms for each edge device to find the best response in games considering the fluence to TSN transmission. In addition, we will jointly allocate the task arrival rates and computing resources (such as computing memory and storage space) to further promote the application of edge computing technology in TSN.

Data Availability

The data used to support the findings of the study are available within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the R&D Project of China under Grant No. 2018YFB1700200 and the Science and Technology Research Program of Chongqing Municipal Education Commission under Grant No. KJQN202000611.

References

[1] L. Lo Bello and W. Steiner, "A perspective on IEEE time-sensitive networking for industrial communication and automation systems," *Proceedings of the IEEE*, vol. 107, no. 6, pp. 1094–1120, 2019.

[2] M. Wollschlaeger, T. Sauter, and J. Jasperneite, "The future of industrial communication: automation networks in the era of the Internet of Things and industry 4.0," *IEEE Industrial Electronics Magazine*, vol. 11, no. 1, pp. 17–27, 2017.

[3] J. Farkas, L. L. Bello, and C. Gunther, "Time-sensitive networking standards," *IEEE Communications Standards Magazine*, vol. 2, no. 2, pp. 20-21, 2018.

[4] Ieee, "Time-sensitive networking task group," 2020, http://www.ieee802.org/1/pages/tsn.html[online].

[5] K. Cao, S. Hu, Y. Shi, A. W. Colombo, S. Karnouskos, and X. Li, "A survey on edge and edge-cloud computing assisted cyber-physical systems," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 11, pp. 7806–7819, 2021.

[6] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[7] M. De Donno, K. Tange, and N. Dragoni, "Foundations and evolution of modern computing paradigms: cloud, IoT, edge, and fog," *IEEE Access*, vol. 7, pp. 150936–150948, 2019.

[8] X. Xia, F. Chen, Q. He, J. C. Grundy, M. Abdelrazek, and H. Jin, "Cost-effective app data distribution in edge computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 1, pp. 31–44, 2021.

[9] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 1, pp. 856–868, 2019.

[10] S. K. U. Zaman, A. I. Jehangiri, T. Maqsood et al., "LiMPO: lightweight mobility prediction and offloading framework using machine learning for mobile edge computing," *Cluster Computing*, 2022.

[11] S. K. u Zaman, A. I. Jehangiri, T. Maqsood et al., "COME-UP: computation offloading in mobile edge computing with LSTM based user direction prediction," *Applied Sciences*, vol. 12, no. 7, p. 3312, 2022.

[12] E. Mustafa, J. Shuja, K. Bilal et al., "Reinforcement learning for intelligent online computation offloading in wireless powered edge networks," *Cluster Computing*, 2022.

[13] Ieee, "802.1Qbv—enhancements for scheduled traffic," Institute of Electrical and Electronics Engineers, 2016, http://www.ieee802.org/1/pages/802.1bv.html[Online].

[14] K. Li, "How to stabilize a competitive mobile edge computing environment: a game theoretic approach," *IEEE Access*, vol. 7, pp. 69960–69985, 2019.

[15] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: a survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, 2018.

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

[17] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems," in *Proceedings of the 2016 IEEE International Symposium on Information Theory (ISIT)*, pp. 1451–1455, Barcelona, Spain, July 2016.

[18] M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 587–597, 2018.

[19] H. Xing, L. Liu, J. Xu, and A. Nallanathan, "Joint task assignment and wireless resource allocation for cooperative mobile-edge computing," in *Proceedings of the 2018 IEEE International Conference on Communications (ICC)*, pp. 1–6, Kansas City, MO, USA, May 2018.

[20] Z. Jiang and S. Mao, "Energy delay tradeoff in cloud offloading for multi-core mobile devices," *IEEE Access*, vol. 3, pp. 2306–2316, 2015.

[21] H. Wu, Y. Sun, and K. Wolter, "Energy-efficient decision making for mobile cloud offloading," *IEEE Transactions on Cloud Computing*, vol. 8, no. 2, pp. 570–584, 2020.

[22] Y. Mao, J. Zhang, S. H. Song, and K. B. Letaief, "Power-delay tradeoff in multi-user mobile-edge computing systems," in *Proceedings of the 2016 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, Washington, DC, USA, December 2016.

[23] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 4, pp. 974–983, 2015.

[24] J. Zheng, Y. Cai, Y. Wu, and X. Shen, "Dynamic computation offloading for mobile cloud computing: a stochastic game-theoretic approach," *IEEE Transactions on Mobile Computing*, vol. 18, no. 4, pp. 771–786, 2019.

[25] K. Li, "A game theoretic approach to computation offloading strategy optimization for non-cooperative users in mobile edge computing," *IEEE Transactions on Sustainable Computing*, p. 1, 2018.

[26] Y. Hui, Z. Su, T. H. Luan, C. Li, G. Mao, and W. Wu, "A game theoretic scheme for collaborative vehicular task offloading in 5G HetNets," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 16044–16056, 2020.

[27] Y. Wang, P. Lang, D. Tian et al., "A game-based computation offloading method in vehicular multiaccess edge computing networks," *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 4987–4996, 2020.

[28] T. Fang, F. Yuan, L. Ao, and J. Chen, "Joint task offloading, D2D pairing, and resource allocation in device-enhanced MEC: a potential game approach," *IEEE Internet of Things Journal*, vol. 9, no. 5, pp. 3226–3237, 2022.

[29] Y. Ding, K. Li, C. Liu, and K. Li, "A potential game theoretic approach to computation offloading strategy optimization in end-edge-cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 6, pp. 1503–1519, 2022.

[30] V. Cardellini, V. De Nitto Persone, V. Di Valerio et al., "A game-theoretic approach to computation offloading in mobile cloud computing," *Mathematical Programming*, vol. 157, no. 2, pp. 421–449, 2016.

[31] S. M. Ross, *Introduction to Probability Models*, Academic Press, Cambridge, MA, USA, 2014.

[32] Ieee, "IEEE standard for local and metropolitan area network--bridges and bridged networks," *IEEE Std 802.1Q-2018 (Revision of IEEE Std 802.1Q-2014)*, pp. 1–1993, 2018.

[33] Y. Shoham and K. Leyton-Brown, *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*, Cambridge University Press, Cambridge, UK, 2008.

[34] Y. H. Chew and B. H. Soong, *Potential Game Theory: Applications in Radio Resource Allocation*, Springer Publishing Company, Berlin, Germany, 2016.