sensors

MDPI

Article

Online Workload Allocation via Fog-Fog-Cloud Cooperation to Reduce IoT Task Service Delay

Lei Li [1], Mian Guo [2], Lihong Ma [1], Huiyun Mao [3] and Quansheng Guan [1,*]

[1] School of Electronic and Information Engineering, South China University of Technology, Guangzhou 510641, China
[2] School of Electronic and Information Engineering, Guangdong University of Petrochemical Technology, Maoming 525000, China
[3] School of Computer Science and Engineering, South China University of Technology, Guangzhou 510641, China
* Correspondence: eeqshguan@scut.edu.cn or qshguan@gmail.com; Tel.: +86-20-87114700

Received: 13 August 2019; Accepted: 2 September 2019; Published: 4 September 2019

Abstract: Fog computing has recently emerged as an extension of cloud computing in providing high-performance computing services for delay-sensitive Internet of Things (IoT) applications. By offloading tasks to a geographically proximal fog computing server instead of a remote cloud, the delay performance can be greatly improved. However, some IoT applications may still experience considerable delays, including queuing and computation delays, when huge amounts of tasks instantaneously feed into a resource-limited fog node. Accordingly, the cooperation among geographically close fog nodes and the cloud center is desired in fog computing with the ever-increasing computational demands from IoT applications. This paper investigates a workload allocation scheme in an IoT–fog–cloud cooperation system for reducing task service delay, aiming at satisfying as many as possible delay-sensitive IoT applications' quality of service (QoS) requirements. To this end, we first formulate the workload allocation problem in an IoT-edge-cloud cooperation system, which suggests optimal workload allocation among local fog node, neighboring fog node, and the cloud center to minimize task service delay. Then, the stability of the IoT-fog-cloud queueing system is theoretically analyzed with Lyapunov drift plus penalty theory. Based on the analytical results, we propose a delay-aware online workload allocation and scheduling (DAOWA) algorithm to achieve the goal of reducing long-term average task serve delay. Theoretical analysis and simulations have been conducted to demonstrate the efficiency of the proposal in task serve delay reduction and IoT-fog-cloud queueing system stability.

Keywords: fog system; internet of things; workload allocation; task service delay; Lyapunov drift-plus-penalty

1. Introduction

The increasing number of Internet of Things (IoT) applications, such as audio recognition, vehicle-to-roadside communications, and virtual reality, often demand a low end-to-end latency between a sensor and a control center [1,2]. These delay-sensitive applications often have stringent task service delay (TSD) requirements, which presents the total delay from the moment at which the task enters the system to when the process is completed.

The development of delay-sensitive IoT applications has presented increasing challenges for the current cloud computing infrastructure. TSD contains not only the computation delay, but also the queuing delay and network delay. Although cloud computing can provide a low-cost, easily expandable, and on-demand high-performance computation service [3–5], it relies on huge volumes of data transmissions from the IoT end devices to the remote cloud center, consuming an extremely

large amount of network bandwidth resources, as well as causing considerable network delay. Cloud computing has become the bottleneck for the development of delay-sensitive IoT applications [6,7].

Fog computing [8,9], which is a middle-tier between the IoT end devices and cloud center, has emerged as a solution to provide high-resilience service quality [10]. It can be considered as an extension of cloud computing, and processes tasks at the edge devices of the network with the aim of preventing the large network delay. Nevertheless, the computation capability of a fog node is limited due to its geographical location (e.g., a pylon or wireless base station) and limited power supply (e.g., solar energy or wind power in remote areas). When the workload of the fog node is heavy, tasks have to be queued at the fog node, and may experience a long queuing delay. In some serious situations, the queuing delay may even exceed the network delay.

Owing to the limited computation capability of a single fog node, it will be useful to explore and exploit the cooperation among multiple fog nodes and the cloud center to provide lower TSDs. The cooperation of multiple closely connected fog nodes provides a geographically local computation coalition. Thus, the data transmission and network delay between the local fogs and the remote cloud center is reduced significantly [7]. The local computation coalition also reduces the task queueing at a single fog node.

Workload allocation among fog nodes and the cloud center is a key technique that affects the TSD in QoS provisioning [11]. It determines where a task is serviced in the fog system, and affects both the queueing delay and network delay. However, the dynamic traffic characteristics, and heterogeneous computation capabilities of fog nodes and the cloud center present many challenges for workload allocation. First, the tasks are generated stochastically and the amount of computation also varies for different tasks and over time [12]. An online algorithm would thus be required to solve a workload allocation. Second, owing to the heterogeneous resources of fog nodes and the cloud center, there exists a tradeoff between the queuing delay and the network delay, which complicates workload allocation for tasks. Offloading more tasks to the cloud will increase the network delay, however will decrease the queueing delay at fogs, and vice versa.

The main focus of this paper is to study the workload allocation problem for an IoT-fog-cloud system with the aim of reducing the TSD. When a task is generated at an IoT end device, it will be delivered to its upstream local fog node. Then, the task can be processed at the local fog node, offloaded to the neighboring fog node, or offloaded to the cloud center. Our goal is to find an optimal workload allocation scheme to allocate workload among the local fog node, neighboring fog nodes, or the cloud center according to the system states, which is formulated as a delay-aware workload allocation problem with the goal of minimizing the TSD. The problem is tackled by a proposed online workload allocation algorithm using Lyapunov drift-plus-penalty theory.

Our main contributions can be summarized as follows:

- Based on the IoT–fog–cloud system architecture, we present a time-varying queuing model that explicitly considers the heterogeneous computational capability and network delay. Then, a delay-based workload allocation problem is formulated, which suggests the optimal workload allocations among local fog node, neighboring fog nodes, and the cloud center to minimize TSD for tasks.
- We apply the Lyapunov optimization method [13,14] to find out a solution of the workload allocation scheme. Specifically, the drift-plus-penalty properties of the TSD minimization with respect to system stabilization are analyzed. Then, a delay-aware online workload allocation and scheduling algorithm, which enables the local fog node to cooperate with neighboring fog nodes and the cloud center, is proposed. The algorithm can optimize the workload allocation to reduce the average TSD according to the system status online.
- Theoretical analysis and simulation evaluations both illustrate that our proposed algorithm achieves a lower TSD compared with other algorithms.

The remainder of the paper is organized as follows. Section 2 discusses related work. Section 3 introduces the system structure and traffic model. Section 4 describes the problem formulation. Details

of the proposed online algorithm are presented in Section 5, in which we also provide a performance analysis of the proposed algorithm. Section 6 presents the simulation evaluation and the results. Finally, Section 7 summarizes the paper.

2. Related Work

Fog computing, which provides a flexible computing paradigm with low delay, high security and high energy efficiency, has received an increasing amount of attention in recent years. One of the popular research fields in fog computing is the development of an offloading policy to determine when/where the task can be offloaded and processed by a suitable device in the fog system (e.g., a fog node or the cloud center).

Xu et al. [15] proposed an online learning algorithm to determine workload offloading in mobile edge computing to minimize the cost of the edge device. Bagula et al. [16] proposed a model for micro-level cost estimation. Based on the model, they proposed a resource allocation algorithm that benefits both the customers and the providers. Similarly, Amoretti et al. [17] proposed a mobile cloud computing simulation model based on queuing network architecture and designed a task offloading policy that optimizes the energy efficiency. Fan and Ansari [18] proposed a workload allocation policy for base stations that considered both the network delay and computing delay to reduce the resource cost and response delay based on an M/M/1 queuing model. Lyu et al. [19] proposed a task offloading policy based on the task delay requirements. Chang et al. [20] also proposed a distributed algorithm based on the alternating direction method of multipliers (ADMM) to determine whether the task should be offloaded to a fog node with the energy efficiency of the user side. Guo et al. [21] provided an energy-efficient dynamic offloading and resource scheduling policy to reduce energy consumption and shorten the application completion time of smart devices in mobile cloud computing. Rahbari and Nickray [22] presented a module placement method based on a classification and regression tree algorithm to offload the task such that the power consumption was minimized.

Additionally, researchers previously focused on the task offloading based on a three-tier fog system model. Similar to the effort of Li et al. [23], Wu et al. [24] also proposed a three-level mathematical model that included the end devices, middleware consisting of fog nodes, and cloud center. Based on the model, a task offloading algorithm was proposed based on the predicted energy consumption. More importantly, this study considered the computation capability of fog node middleware to be larger than that of the end devices, but smaller than that of the cloud center. Ma et al. [25] proposed an IoT-based fog computing model. Based on the model, a genetic algorithm was proposed for reducing the failure node and energy consumption. Yousefpour et al. [26] proposed a mathematical model of a three-tier fog system to evaluate its performance. They used a threshold method as the task offloading decision to reduce the task delay. Nan et al. [27] also used a queuing model to analyze the performance of the Cloud of Things (CoTs) system consisting of end devices, fog nodes, and a cloud center. They proposed a task offloading policy based on the Lyapunov optimization to minimize the energy cost. Deng et al. [28] also formulated a workload allocation solution that suggested optimal workload allocations between the fog and the cloud and minimizes the power consumption with constrained service delay. Nawrocki and Reszelewski [29] presented two types of offloading task resource usage in a mobile cloud system. Their experimental results showed that resource utilization using the multiple user-one virtual machine (VM) mode was higher than that using the one user-one VM mode, but the performance of the former was lower than that of the latter. Compared with the three-tier fog system, the resource usage of the fog node tier was similar to that of the multiple user-one VM mode because of the limited computation resource, but the cloud center could use the one user-one VM mode to improve the task delay.

This study differs from existing work in the following respects. In this study, workload allocation in a, three-tier fog system (i.e., IoT-fog-cloud fog system) is studied with dynamic workloads, where the computation capability and complicated network delay in different tiers of the system are considered. Thus, we need to find out an online workload allocation scheme among the local fog node, neighboring

fog nodes and the cloud center to minimize the task service delay. Then, we propose a fine granular low-complexity workload allocation scheme, which can adaptively switch among the local fog node, neighboring fog nodes, and the cloud center for workload allocation according the system status online. To the best of our knowledge, this is the first effort designed to attain optimal workload allocation for minimal per-task granular service delay in such a three-tier fog system.

3. System Description

3.1. Internet of Things (IoT)-Fog-Cloud System Model

As shown in Figure 1, this paper considers an IoT–fog–cloud system, which is divided into three tiers [23] as shown in Figure 1. The end tier consists of multiple IoT devices distributed in several geographically adjacent regions (called IoT regions). The fog nodes form a fog network (f2f network) in the fog tier, where each fog node mainly provides computing services to one downstream IoT region. Thus, a fog node is called the local fog node of the corresponding downstream IoT region [27]. The cloud tier includes the cloud center. A computation task generated in an IoT device can either be computed locally in its upstream fog node, or be offloaded to a neighboring fog through the f2f network, or be offloaded to the cloud center through the fog-to-cloud (f2c) network.

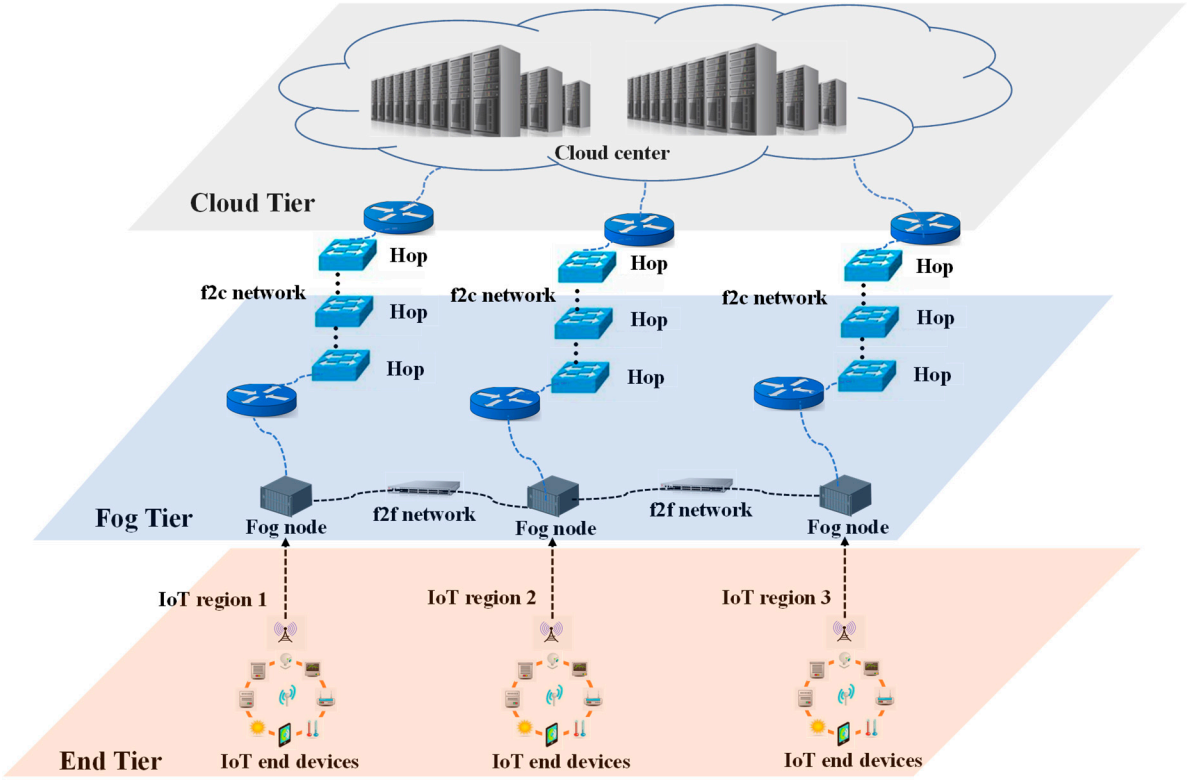

Figure 1. Internet of things (IoT)-fog-cloud system architecture.

Note that, when a task is offloaded to a neighboring fog node or the cloud center, it results in three types of network delays, including propagation, transmission, and congestion delays. The propagation delay, which is caused by multi-hop transmissions among routers and switches, can be obtained by using the PING command. The longer the distance is, the longer propagation delay is. The transmission delay is caused by the limited availability of network bandwidth when data is transmitted via the network. When the amount of data to be transmitted increases, the data cache in the network device also increases. Thus, if the amount of data to be transmitted exceeds the bandwidth resource, the remaining data in the

network device will introduce the congestion delay in the next time slot. The processing flowchart of the fog system based on the description above is shown in Figure 2.

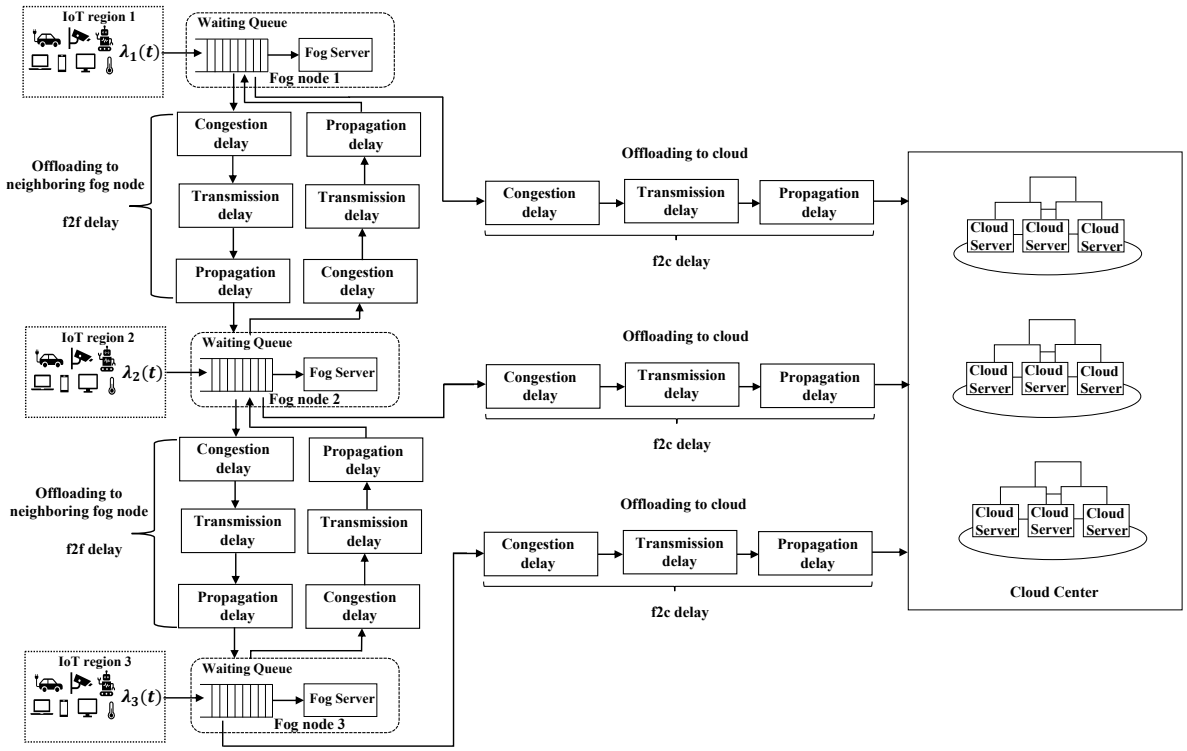

Figure 2. Flow chart showing the processes in the IoT-fog-cloud system.

Assume there are R IoT regions and R fog nodes. Let $\mathcal{R} = \{1, \ldots, R\}$. Assume that fog node j's downstream IoT region is IoT region j, for $\forall j \in \mathcal{R}$. Let F_j^{fog} $(j \in \mathcal{R})$ be the CPU cycle frequency of fog node j. Then, the computation speed P_j, measuring in million instructions per second (MIPS) [30], is derived by $P_j = \frac{F_j^{fog}}{10^6 \times CPI}$, (where CPI means clock cycle per instruction). The computation resource of the cloud center is assumed to be unlimited in comparison with a fog node, such that any task can be processed immediately after its arrival. Any task is allocated a computation speed $P_c = \frac{F^{cloud}}{10^6 \times CPI}$ immediately after its arrival, where F^{cloud} is the CPU cycle frequency of the cloud center allocated to the task. Assume that $P_j < P_c$.

3.2. Traffic Model

A dynamic discrete-time IoT-fog-cloud system is considered [31]. The arrival process of workloads is as follows: (1) at every time slot, tasks are generated from each IoT region stochastically and independently; (2) in each IoT region, the generated number of tasks per time slot follows an independent and identical distribution (i.i.d); (3) the task length (in million instructions (MI)) and data size (in bits) of each task follow the i.i.d, respectively.

Let $S_{(i,j)}^{(t)}$ be the i^{th} task that is generated from IoT region j in time slot t. The task $S_{(i,j)}^{(t)}$ is modeled as $\left\{l_{(i,j)}^{(t)}, d_{(i,j)}^{(t)}\right\}$, where $l_{(i,j)}^{(t)}$ and $d_{(i,j)}^{(t)}$ present the task length and data size, respectively.

Let $\chi_j(t)$ be the task space containing the tasks generated from IoT region j in time slot t. Let $\lambda_j(t)$ denote the number of tasks in task space $\chi_j(t)$. Then, $Xw_j(t) = \sum_{i \in \chi_j(t)} l_{(i,j)}^{(t)}$ and $Ys_j(t) = \sum_{i \in \chi_j(t)} d_{(i,j)}^{(t)}$ represent the accumulative computation workloads and data sizes from IoT region j in time slot t, respectively. Let $\overline{\lambda_j(t)} = E\left[\lambda_j(t)\right]$ be the average task generation rate in IoT region j in time slot t, and

the long-term average task generation rate is $\overline{\lambda_j} = \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \overline{\lambda_j(t)}$. Let $\overline{l_j(t)} = E\left[\sum_{i=1}^{\lambda_j(t)} l_{(i,j)}^{(t)} / \lambda_j(t)\right]$ and $\overline{d_j(t)} = E\left[Ys_j(t)/\lambda_j(t)\right]$ be the expected task length and expected data size in IoT region j in time slot t, respectively. The corresponding long-term expected task instruction length and expected data size of tasks generated in IoT region j are $\overline{l_j} = \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \overline{l_j(t)}$ and $\overline{d_j} = \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \overline{d_j(t)}$, respectively.

Let $\chi_j^j(t), \chi_j^{(k)}(t)$ and $\chi_j^c(t)$ be the task space containing the tasks that are determined to be processed at the fog node j in time slot t (i.e., local fog node, $j \in R$), offloaded to neighboring fog node k, and offloaded to the cloud center in time slot t, respectively. $N_j^j(t)$, $N_j^{(k)}(t)$, and $N_j^c(t)$ are the corresponding number of tasks in task spaces $\chi_j^j(t)$, $\chi_j^{(k)}(t)$, and $\chi_j^c(t)$, respectively. Further, $Xw_j^j(t) = \sum_{i\in\chi_j^j(t)} l_{(i,j)}^{(t)}$, $Xw_j^{(k)}(t) = \sum_{i\in\chi_j^{(k)}(t)} l_{(i,j)}^{(t)}$, and $Xw_j^c(t) = \sum_{i\in\chi_j^c(t)} l_{(i,j)}^{(t)}$ are the corresponding workloads, respectively. We use $\mathcal{K}$ to represent the total space of the neighboring fog nodes of fog node j. Thus, $\chi_j^K(t) = \cup_{k\in\mathcal{K}} \chi_j^{(k)}(t)$ is the total task space containing the tasks that are determined to be offloaded to the neighboring fog nodes of fog node j in time slot t. The corresponding number of tasks in task space $\chi_j^K(t)$ is $N_j^K(t) = \sum_{k\in\mathcal{K}} N_j^{(k)}(t)$. The total corresponding workloads that are determined to be allocated to the neighboring fog nodes of fog node j are $Xw_j^K(t) = \sum_{k\in\mathcal{K}} Xw_j^{(k)}(t)$. Additionally, $\chi_j(t) = \chi_j^j(t) \cup \chi_j^K(t) \cup \chi_j^c(t)$.

Since the tasks from IoT region j will finally be processed in the local fog node, or the neighboring fog nodes, or the cloud center, we have:

$$\begin{cases} \lambda_j(t) = N_j^j(t) + N_j^K(t) + N_j^c(t), \\ Xw_j(t) = Xw_j^j(t) + Xw_j^K(t) + Xw_j^c(t). \end{cases} \tag{1}$$

3.3. Delay Model

3.3.1. Task Service Delay (TSD)

As shown in Figure 2, when the task $S_{(i,j)}^{(t)}$ is generated in IoT region j and delivered to the local fog node in time slot t, its TSD is determined by the workload allocation decision. Besides the computation delay, if the task $S_{(i,j)}^{(t)}$ is offloaded to the neighboring fog node or the cloud center, the task transmission through the network will cause the network delay. Let $Ct(t)_{(i,j)}^{(m)}$ ($m \in \{j\} \cup \mathcal{K} \cup \{c\}$), $Tf_{(i,j)}^{(k)}(t)$ and $Tc_{(i,j)}(t)$ be the computation delay, f2f network delay, and f2c network delay, respectively. Thus, the TSD of the task $S_{(i,j)}^{(t)}$ can be obtained as:

$$\begin{aligned} T_{(i,j)}(t) = I_{(i,j)}^j(t)\left(Ct(t)_{(i,j)}^{(j)}\right) + \sum_{k\in\mathcal{K}}\left[I_{(i,j)}^{(k)}(t)\cdot\left(Tf_{(i,j)}^{(k)}(t) + Ct(t)_{(i,j)}^{(k)}\right)\right] \\ + I_{(i,j)}^c(t)\cdot\left(Tc_{(i,j)}(t) + Ct(t)_{(i,j)}^{(c)}\right), \end{aligned} \tag{2}$$

where $I_{(i,j)}^j(t) + \sum_{k\in\mathcal{K}} I_{(i,j)}^{(k)}(t) + I_{(i,j)}^c(t) = 1$. When the task $S_{(i,j)}^{(t)}$ is determined to be processed at the local fog node (i.e., fog node j), $I_{(i,j)}^j(t) = 1$; otherwise, $I_{(i,j)}^j(t) = 0$; when the task $S_{(i,j)}^{(t)}$ is determined to be offloaded to neighboring fog node k, $I_{(i,j)}^{(k)}(t) = 1$; otherwise, $I_{(i,j)}^{(k)}(t) = 0$; when the task $S_{(i,j)}^{(t)}$ is determined to be offloaded to the cloud center, $I_{(i,j)}^c(t) = 1$; otherwise, $I_{(i,j)}^c(t) = 0$. The details of the computation delay and network delay are described as follows.

3.3.2. Computation Delay

Since the computation capability of a fog node is far less than that in the cloud, queueing may happen in a fog node when the workload is heavy. Therefore, we model the computation delays for the fog nodes and the cloud center respectively as follows:

(1) Computation delay at a fog node:

Owing to the limited computation capability of the fog node, we assume that each fog node is a queuing subsystem for buffering the tasks. Let $Q_j(t)$ be the number of tasks being queued in the subsystem of fog node j in time slot t. Assume $Q_k(0) = 0$, and based on Equation (1), the queue length Q_j is evaluated as follows:

$$Q_j(t+1) = \max\Big[Q_j(t) + N_j^e(t) + N_j^j(t) - \mu_j(t), 0\Big], \tag{3}$$

where $N_j^e(t)$ and $N_j^j(t)$ are the number of tasks offloaded from neighboring fog nodes and from downstream IoT region respectively that are determined to be processed at fog node j in time slot t. $\mu_j(t)$ is the number of tasks that are finished at fog node j in time slot t.

Let $A_j(t)$ be the total number of tasks that are determined to be processed at fog node j in time slot t, i.e., $A_j(t) = N_j^e(t) + N_j^j(t)$. Thus, Equation (2) can be rewritten as follows:

$$Q_j(t+1) = \max\Big[Q_j(t) + A_j(t) - \mu_j(t), 0\Big]. \tag{4}$$

Let $\Lambda_j(t)$ be the corresponding task space of $A_j(t)$. Thus, $Xw_j^A(t) = \sum_{(i,n)\in\Lambda_j(t)} l_{(i,n)}^{(t)}$ is the corresponding workload determined to be allocated to fog node j in time slot t. Let ϕ be the length of the time slot. Thus, $P_j{\cdot}\phi$ is the number of instructions processed at fog node j in time slot t.

Let $Qw_k(t)$ be the corresponding workload considering the number of tasks as well as the length of the queues at fog node j in time slot t. Based on Equation (4), we have

$$Qw_j(t+1) = \max\Big[Qw_j(t) + Xw_j^A(t) - P_j{\cdot}\phi, 0\Big]. \tag{5}$$

Based on Equation (5), if the task $S_{(i,m)}^{(t)}$ is allocated to fog node j in time slot t, its queuing delay can be evaluated as follows:

$$Wq_{(i,m)}^{(j)}(t) = \frac{Qw_j(t) + \sum_{(l.n)\in\Gamma_{(i,m)}} l_{(l,n)}^{(t)}}{S_j}, \tag{6}$$

where $\Gamma_{(i,m)}(t)$ is the task space containing the tasks that enter into fog node j before the task $S_{(i,m)}^{(t)}$ in time slot t. Additionally, when the computation resource becomes available to the task $S_{(i,m)}^{(t)}$, its execution time can be calculated as follows:

$$Te_{(i,m)}^{(j)} = \frac{l_{(i,m)}^{(t)}}{P_j}. \tag{7}$$

Accordingly, the computation delay of the task $S_{(i,m)}^{(t)}$ at fog node j can be evaluated as follows:

$$Ct(t)_{(i,m)}^{(j)} = Wq_{(i,m)}^{(j)}(t) + Te_{(i,m)}^{(j)}. \tag{8}$$

(2) Computation delay in the cloud center:

By contrast, since any task can be processed immediately after its arrival, the computation delay of the task $S_{(i,m)}^{(t)}$ at the cloud center equals to the execution time, which can be calculated as follows:

$$Ct(t)_{(i,m)}^{c} = Te_{(i,m)}^{c} = \frac{l_{(i,m)}^{(t)}}{P_c}. \tag{9}$$

3.3.3. Network Delay

As shown in Figure 2, there are two types of network transmission paths: the f2f and f2c network paths. We model these two types of network delay as follows, respectively.

(1) f2f delay:

Let $\left\{Bw_j^{(k)}, Tp_j^{(k)}\right\}$ denote the parameters of the network that transmits the data from fog node j to fog node k, where $Bw_j^{(k)}$, $Tp_j^{(k)}$ represent the bandwidth and the propagation delay, respectively.

Since the f2f network is generally a one-hop and bandwidth-constrained network in comparison with the f2c network, network congestion may happen in the f2f network in a traffic-bursting period. We use $G_j^{(k)}(t)$ to represent the remaining data to be transmitted in this link at the beginning of time slot t. Specially, $G_j^{(k)}(t) = 0$ indicates no congestion. Thus, the congestion delay contributing to the f2f delay is calculated as:

$$Tg_{(i,j)}^{(k)}(t) = \frac{G_j^{(k)}(t) + \sum_{n \in Z_{(j)}^{(k)}(t)} d_{(n,j)}^{(t)}}{Bw_j^{(k)}}, \tag{10}$$

where $Z_{(j)}^{(k)}(t)$ contains the tasks that will be transmitted from fog node j to fog node k before the task $S_{(i,j)}^{(t)}$ in time slot t.

The transmission delay of the task $S_{(i,j)}^{(t)}$ can be calculated as follows:

$$Ts_{(i,j)}^{(k)}(t) = \frac{d_{(i,j)}^{(t)}}{Bw_j^{(k)}}. \tag{11}$$

Therefore, if the task $S_{(i,j)}^{(t)}$ is transmitted from fog node j to fog node k, its f2f delay can be evaluated as follows:

$$Tf_{(i,j)}^{(k)}(t) = Tg_{(i,j)}^{(k)}(t) + Ts_{(i,j)}^{(k)}(t) + Tp_j^{(k)}. \tag{12}$$

(2) f2c delay:

Let $\left\{B_j^c, Tp_j^c\right\}$ represent the bandwidth and propagation delay of the f2c. Since the network resource of the datacenter network is far sufficient in comparison with the edge network (e.g., f2f network), it is reasonable to assume that $Bw_j^k < Bw_j^c$ and $Tp_j^{(k)} < Tp_j^c$ [32]. Similar to the propagation model in Section 3.3.2-(1), we use again $G_j^c(t)$ to represent the remaining data to be transmitted in the link from fog node j to the cloud center at the beginning of time slot t. Thus, the congestion delay can be calculated as:

$$Tg_{(i,j)}^{c}(t) = \frac{G_j^c(t) + \sum_{n \in Z_{(j)}^c(t)} d_{(n,j)}^{(t)}}{Bw_j^c}, \tag{13}$$

where $Z_{(j)}^c(t)$ contains the tasks that will be transmitted from fog node j to the cloud center before the task $S_{(i,j)}^{(t)}$. The transmission delay can be calculated as follows:

$$Ts^{c}_{(i,j)}(t) = \frac{d^{(t)}_{(i,j)}}{Bw^{c}_{j}}. \tag{14}$$

Thus, if the task $S^{(t)}_{(i,j)}$ is offloaded from fog node j to the cloud center, its f2c delay can be evaluated as follows:

$$Tc_{(i,j)}(t) = Tg^{c}_{(i,j)}(t) + Ts^{c}_{(i,j)}(t) + Tp^{c}_{j}. \tag{15}$$

3.3.4. Average Task Service Delay

In a lossless system, based on Equation (2), the average TSD of tasks generated from all regions in time slot t is derived by:

$$T_{avg}(t) = \frac{\sum_{j=1}^{R}\sum_{i=1}^{\lambda_j(t)} T_{(i,j)}(t)}{\sum_{j=1}^{R}\lambda_j(t)}. \tag{16}$$

The long-term average TSD of tasks generated from all regions can be calculated by:

$$\overline{T_{avg}} = \lim_{T\to\infty}\frac{1}{T}\sum\nolimits_{t=0}^{T-1} E\Big[T_{avg}(t)\Big]. \tag{17}$$

4. Problem Formulation and Transformation

4.1. Problem Formulation

Our goal is to reduce the value of $\overline{T_{avg}}$ in Equation (17). At the same time, to avoid an extremely long queuing delay, the queuing system should be stable, where the stability of the fog system is defined as follows.

Definition 1. *(Stability of a fog system). A fog system is stable if queue vector* $\boldsymbol{Q}(t)$ *and workload vector* $\boldsymbol{Qw}(t)$ *are both stable, where* $\boldsymbol{Q}(t) = [Q_1(t), Q_2(t), \ldots, Q_R(t)]$ *and* $\boldsymbol{Qw}(t) = [Qw_1(t), Qw_2(t), \ldots, Qw_R(t)]$, respectively, i.e.,

$$\begin{cases} \lim\limits_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1} E\Big[\sum_{j=1}^{R} Q_j(t)\Big] < \infty \\ \lim\limits_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1} E\Big[\sum_{j=1}^{R} Qw_j(t)\Big] < \infty \end{cases}. \tag{18}$$

Let $\pi_{(i,j)}(t) = \begin{pmatrix} I^{j}_{i}(t) & \ldots & I^{(k)}_{i}(t) & \ldots & I^{c}_{i}(t) \end{pmatrix}$ be the decision vector for the task $S^{(t)}_{(i,j)}$. Then, the workload allocation decisions for $\lambda_j(t)$ arriving tasks at time slot t at fog node j can be represented as follows:

$$\pi^{(j)}(t) = \Big(\pi_{(1,j)}(t), \pi_{(2,j)}(t), \ldots, \pi_{(\lambda_j(t),j)}(t)\Big). \tag{19}$$

The decision vector for all tasks generated from all fog nodes is expressed as:

$$\boldsymbol{\pi}(t) = \Big(\pi^{(1)}(t), \ldots, \pi^{(j)}(t), \ldots, \pi^{(R)}(t)\Big). \tag{20}$$

Then, according to the definitions of $N^{j}_{j}(t)$, $N^{(k)}_{j}(t)$, and $N^{c}_{j}(t)$ as well as $Xw^{j}_{j}(t)$, $Xw^{(k)}_{j}(t)$ and $Xw^{c}_{j}(t)$, and $Xw^{c}_{j}(t)$ in Equation (1), we have

$$\begin{cases} N^{j}_{j}(t) = \sum_{i\in\chi_j(t)} I^{j}_{(i,j)}(t), \\ N^{(k)}_{j}(t) = \sum_{i\in\chi_j(t)} I^{(k)}_{(i,j)}(t), \\ N^{c}_{j}(t) = \sum_{i\in\chi_j(t)} I^{c}_{(i,j)}(t). \end{cases} \tag{21}$$

and

$$\begin{cases} Xw_j^j(t) = \sum_{i\in\chi_j(t)} I_{(i,j)}^j(t)\cdot l_{(i,j)}^{(t)}, \\ Xw_j^{(k)}(t) = \sum_{i\in\chi_j(t)} I_{(i,j)}^{(k)}(t)\cdot l_{(i,j)}^{(t)}, \\ Xw_j^c(t) = \sum_{i\in\chi_j(t)} I_{(i,j)}^c(t)\cdot l_{(i,j)}^{(t)}. \end{cases} \tag{22}$$

Since, $E\left[T_{avg}(t)\right]$ in Equation (17) is determined by $\boldsymbol{\pi}(t)$, $E\left[T_{avg}(t)\right]$ can be represented as $E\left[T_{avg}(\boldsymbol{\pi}(t))\right]$. Therefore, the workload allocation problem for minimizing the TSD in the fog system can be formulated as:

$$\begin{aligned} Minimize\overline{T_{avg}} &= \lim_{T\to\infty}\tfrac{1}{T}\sum_{t=0}^{T-1} E\left[T_{avg}(\boldsymbol{\pi}(t))\right] \\ &= \lim_{T\to\infty}\tfrac{1}{T}\sum_{t=0}^{T-1}\frac{\sum_{j=1}^{R}\sum_{i=1}^{\lambda_j(t)} T_{(i,j)}(t)}{\sum_{j=1}^{R}\lambda_j(t)},\ \forall t\in\{0,1,2,\ldots,\infty\}, \\ &s.t.\ \text{Equations (1), (18), (21) and (22).} \end{aligned} \tag{23}$$

where Equation (23) follows Equation (17); Equation (18) is the stability constraint of the system, Equation (1) is the traffic constraint; Equations (21) and (22) follow the definition of $\boldsymbol{\pi}(t)$.

The above problem is equivalent to determining a sequential optimal $\boldsymbol{\pi}^*(t)$ for $t = 0, 1, \ldots, \infty$ to achieve the objective of minimizing $E[\cdot]$, where $\boldsymbol{\pi}^*(t) = \left(\pi^{(1)*}(t), \ldots, \pi^{(j)*}(t), \ldots, \pi^{(R)*}(t)\right)$.

4.2. Problem Transformation

To achieve the objective in Equation (23), we can transfer the problem by minimizing the average TSD in each time slot. Then, the problem in Equation (23) can be transformed as follows:

$$\begin{aligned} &Minimize\ E\left[T_{avg}(\boldsymbol{\pi}(t))\right],\ \forall t\in\{0,\ 1,\ 2,\ldots,\infty\}, \\ &s.t.\ \text{Equations (1), (18), (21) and (22).} \end{aligned} \tag{24}$$

Based on Definition 1 in Section 4.1, we have the following lemma.

Lemma 1. *$\boldsymbol{Q}(t)$ and $\boldsymbol{Qw}(t)$ are both stable if $\overline{Qw_j} < \infty$, where $\overline{Qw_j}$ is the long-term average length of $Qw_j(t)$.*

Proof. According to $\overline{Qw_j} < \infty$, we assume that $\overline{Qw_j} < C < \infty$, where C is a finite constant. Then, we have:

$$\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1} E\left[\sum_{j=1}^{R} Qw_j(t)\right] = \sum_{j=1}^{R} E[\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1} Qw_j(t)] < \sum_{j=1}^{R}\overline{Qw_j} < \sum_{j=1}^{R} C < \infty.$$

Thus, $\boldsymbol{Qw}(t)$ can remain stable if $\overline{Qw_j} < \infty$. Let $\overline{l^j}$ denote the long-term average length of an instruction of a task processed by fog node j. If $\overline{Qw_j} < \infty$, the length of a task instruction is finite. Further, in the system, there is at least one task's instruction length exceeds 0; thus, the average length of a task instruction is larger than 0. Thus, we have $0 < L < \overline{l^j} < \infty$, where L is a finite constant greater than 0. We obtain:

$$\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1} E\left[\sum_{j=1}^{R} Q_j(t)\right] = \sum_{j=1}^{R}\frac{E\left[\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1} Qw_j(t)\right]}{\overline{l^j}} \le \sum_{j=1}^{R}\frac{C}{L} < \infty.$$

Hence, $\boldsymbol{Q}(t)$ and $\boldsymbol{Qw}(t)$ can remain stable if $\overline{Qw_j} < \infty$, which proves Lemma 1. □

Using Lemma 1, the problem in Equation (24) can be transformed as:

$$Minimize\ E\Big[T_{avg}(\boldsymbol{\pi}(t))\Big] = \frac{\sum_{j=1}^{R}\sum_{i=1}^{\lambda_j(t)} T_{(i,j)}(t)}{\sum_{j=1}^{R}\lambda_j(t)},\ \forall t \in \{0,1,2,\ldots,\infty\}, \tag{25}$$

$$s.t.\ \overline{Qw_j} < \infty, \forall j \in \{1,2,\ldots,R\},\ \text{Equations (1), (21) and (22).}$$

5. Delay-Aware Workload Allocation and Task-Scheduling Scheme

5.1. Lyapunov Drift-Plus-Penalty

As mentioned above, to avoid extremely long queuing delay at the fog node, the workload allocation policy needs to ensure that the queuing system of the fog nodes remains stable. The Lyapunov optimization [13,14], which is central to the study of the optimal control in queuing networks, has been used extensively in control theory to ensure the stability of different forms of systems. We again use the Lyapunov optimization technique to find an efficient online workload offloading scheme to maintain the stability of the queuing fog system. Based on Equation (5), the Lyapunov function of the fog nodes in the system is expressed as follows:

$$L(t) = \frac{1}{2}\sum_{j=1}^{R}\Big[Qw_j(t)\Big]^2. \tag{26}$$

The one-step conditional Lyapunov drift, which represents the difference in the Lyapunov function in two consecutive time slots, is:

$$\Delta L(t) = E[L(t+1) - L(t)\big|\boldsymbol{Qw}(t)], \tag{27}$$

where $\boldsymbol{Qw}(t) = [Qw_1(t),\ Qw_2(t),\ldots,Qw_R(t)]$. We have the following theorem.

Theorem 1. *In every time slot t, for any value of $Qw(t)$, and under any policy, the Lyapunov drift of $Qw(t)$* satisfies:

$$\Delta L(t) \le B + E\Big[\sum_{j=1}^{R}\Big(Qw_j(t)\cdot\Big(Xw_j^A(t) - P_j\cdot\phi\Big)\Big)\Big|\boldsymbol{Qw}(t)\Big], \tag{28}$$

where B is a finite constant.

The proof is described in Appendix A.

Our goal is to determine a sequential optimal workload allocation decision $\boldsymbol{\pi}^*(t)$ for $t = 0,1,\ldots,\infty$ to achieve the objective in Equation (25). Following the drift-plus-penalty technique, we can minimize the upper bound of the following expression in each time slot to optimize and stabilize all queues,

$$\Delta L(t) + V\cdot E\Big[T_{avg}(\boldsymbol{\pi}(t))\Big|\boldsymbol{Qw}(t)\Big], \tag{29}$$

where V is a non-negative control parameter that is chosen as desired and affects the queuing workload and the TSD tradeoff.

Accordingly, we add $E\Big[T_{avg}(\boldsymbol{\pi}(t))\Big]$ as penalty to both sides of the Lyapunov drift in Equation (28) as follows:

$$\begin{gathered}\Delta L(t) + V\cdot E\Big[T_{avg}(\boldsymbol{\pi}(t))\Big|\boldsymbol{Qw}(t)\Big] \le \\ B + E\Big[\sum_{j=1}^{R}\Big(Qw_j(t)\cdot\Big(Xw_j^A(t) - P_j\cdot\phi\Big)\Big)\Big|\boldsymbol{Qw}(t)\Big] + V\cdot E\Big[T_{avg}(\boldsymbol{\pi}(t))\Big|\boldsymbol{Qw}(t)\Big].\end{gathered} \tag{30}$$

5.2. Delay-Aware Online Workload Allocation and Task-Scheduling Algorithm

Furthermore, according to the Lyapunov drift theory, if the Lyapunov drift-plus-penalty in Equation (29) is close to zero, or even a negative value, this implies that the queue length would be stabilized, and the TSD would be reduced. Thus, based on Equation (30), the optimization problem can be formulated as minimizing a bound on the following drift-plus-penalty:

$$Minimize E\Big[\sum_{j=1}^{R}\Big(Qw_j(t)\cdot\Big(Xw_j^A(t)-P_j\cdot\phi\Big)\Big)\Big|\boldsymbol{Qw}(t)\Big]+V\cdot E\Big[T_{avg}(\boldsymbol{\pi}(t))\Big|\boldsymbol{Qw}(t)\Big]. \tag{31}$$

Then, Equation (31) can be rewritten as follows:

$$Minimize E\Big[\sum_{j=1}^{R}\Big(Qw_j(t)\cdot\Big(Xw_j^A(t)-P_j\cdot\phi\Big)\Big)+V\cdot T_{avg}(\boldsymbol{\pi}(t))\Big|\boldsymbol{Qw}(t)\Big]. \tag{32}$$

Note that, although $P_j\ \phi$ in Equation (32) can affect $Qw^j(t)$, it is independent from the workload allocated to the fog node. Then, we can transform the problem in Equation (32) as follows:

$$Minimize\ \sum_{j=1}^{R}\Big[Qw_j(t)\cdot Xw_j^A(t)\Big]+V\cdot T_{avg}(\boldsymbol{\pi}(t)). \tag{33}$$

Two methods are available to achieve the objective in Equation (33): the central management framework and the distributed management framework. Central management relies on a central node to manage the workload allocation of all fog nodes in the fog system. However, the central node requires information about the system status. Thus, the central node needs to traverse all fog nodes to determine the workload allocation of the tasks, which would cause frequent information broadcasts about the system status. Because of the need for the traversal operation involving all fog nodes, it would be difficult to complete the frequent broadcasts in one time slot. Similarly, the workload allocation command sent from the central node to the other fog nodes cannot be guaranteed in real time. Furthermore, the delay incurred when sending the command and the traversal of all fog nodes also increases the task-waiting delay. This indicates that the central management framework is difficult to be deployed and used in a real situation.

Thus, we propose a distributed management framework to find out the solution. Specifically, based on Equation (33), we propose the delay-aware online workload allocation (DAOWA) algorithm to determine the sequential optimal workload allocations $\pi^{(j)*}(t)$ for fog node j for $t = 0, 1, \ldots, \infty$ to minimize the drift-plus-penalty in every time slot, such that we achieve the goal of stabilizing the queue length and reducing the TSD. The pseudocode of the DAOWA algorithm is detailed in Algorithm 1, which follows:

Each fog node broadcasts its own status information to its neighboring fog nodes immediately after all processing schemes are put into operation.

(1) In time slot t, to operate the DAOWA algorithm, each fog node needs to evaluate the information of its neighboring fog nodes based on information that was previously broadcasted. The fog node updates the evaluation only if it receives the newly broadcasted information.
(1) In each time slot, each fog node manages its own workload allocation independently based on the evaluation of its own status information and the neighboring fog node status information.

Furthermore, in a distributed management framework, because each fog node is responsible for its own workload allocation, we use Equation (34) instead of Equation (33) in Algorithm 1. Each fog node can only obtain the information (e.g., $Qw(t)$ or the workload allocation decision) of its neighboring fog nodes at most once in a slot; thus, fog node j uses $Xw_j^j(t)$ and $Xw_j^{(k)}(t)$ in Equations (35) and (36) to approximate the workload of the local fog node and neighboring fog node k in time slot t, respectively.

Similarly, fog node j uses the average TSD of tasks generated from the local region in time slot t, i.e., $T_{avg}^{(j)}(t)$ ($T_{avg}^{(j)}(t) = \frac{\sum_{i=1}^{\lambda_j(t)} T_{(i,j)}(t)}{\lambda_j(t)}$) in Equation (34), to approximate $T_{avg}(\boldsymbol{\pi}(t))$.

Algorithm 1 Delay-aware online workload allocation (DAOWA) algorithm

Input: $\left[Qw_j(t), \ldots, Qw_k(t), \ldots\right]$ and $\left[Xw_j^A(t), \ldots, Xw_k^A(t), \ldots\right]$, $\forall j \in \mathcal{R}, k \in \mathcal{K}, \mathcal{K} \in \mathcal{R}$ and $j \neq k$, where $\mathcal{R}$ and $\mathcal{K}$ are the fog node space and the space of the neighboring fog node of fog node j, respectively.

1) Initialization:

$Xw_j^A(t) = 0, \ldots, Xw_k^A(t) = 0, \ldots$.

2) Decision process:

For task arrival in time slot t, **do**

Choose $\pi^{j*}(t)$ as the optimal decision for fog node j as follows:

$$Minimize \sum\nolimits_{m \in \{j, \mathcal{K}\}} \left[Qw_m(t) \cdot Xw_m^A(t)\right] + V \cdot T_{avg}^{(j)}(t) \tag{34}$$

s.t. (1), (21), (22),

$$Xw_j^A(t) = Xw_j^j(t) = \sum\nolimits_{\chi_j(t)} I_{(i,j)}^j(t) \cdot l_{(i,j)}^{(t)}, \tag{35}$$

$$Xw_k^A(t) = Xw_j^{(k)}(t) = \sum\nolimits_{i \in \chi_j(t)} I_{(i,j)}^{(k)}(t) \cdot l_{(i,j)}^{(t).} \tag{36}$$

3) Processing the decisions:

Observer $\pi_{(i,j)}(t)$ in $\pi^{(j)*}(t)$, **do**

a) **If** $I_{(i,j)}^j(t) = 1$, **do**

Buffer the task $S_{(i,j)}^{(t)}$ into the local fog node (i.e., fog node j);

b) **Else if** $I_{(i,j)}^{(k)}(t) = 1$, **do**

Transmit the task $S_{(i,j)}^{(t)}$ to neighboring fog node k;

c) **Else if** $I_{(i,j)}^c(t) = 1$, **do**

Transmit the task $S_{(i,j)}^{(t)}$ to the cloud center.

Output: $\pi^{j*}(t)$.

In addition to the workload allocation process, the task undergoes a scheduling process in the fog system. The workload allocation process determines where to process the task, whereas the scheduling process services the task based on the workload allocation decision. Therefore, based on Algorithm 1, we proposed an online workload allocation and task-scheduling algorithm, namely DAOWA-based workload allocation and task-scheduling algorithm, which is described in Algorithm 2.

Algorithm 2 DAOWA-based workload allocation and task scheduling algorithm

1) Initialization:

For each fog node, **do**

$Qw_j(0) = 0, \ldots, Qw_k(t) = 0, \ldots, \forall j \in \mathcal{R}, k \in \mathcal{K}$, and $j \neq k$, where $\mathcal{R}$ and $\mathcal{K}$ are the fog node space and the space of the neighboring fog node of fog node j, respectively.

2) The task workload allocation process:

For each time slot, **do**

a) **For** all fog nodes in parallel:

i) **Initialization:**

Update the queuing status evaluation of the neighboring fog nodes, i.e., $Qw_k(t), \forall k \in K$, according to the previous broadcast.

ii) **Workload allocation process:**

Run Algorithm 1 for fog node j to obtain $\pi^{(j)*}(t)$.

3) Task scheduling process:

For each time slot, **do**

a) **For** all fog nodes in parallel:

i) Schedule the workload of task according to $\pi^{(j)*}(t)$ which is obtained from Algorithm 1;

ii) Process the waiting tasks with $P_j \cdot \phi$ in the first-in-first-out (FIFO) discipline;

iii) **If** the fog node receives tasks through the f2f network transmission, **do**

Buffer the tasks in the waiting queue of the node in the j^{th} region.

Update $Qw_j(t)$, $Q_j(t)$ and congestion events with Equations (4) and (5).

b) **For** the cloud center:

If the cloud center receives tasks via f2c network transmission, **do**

Initiate the VMs with the same number of tasks to process the tasks.

4) Broadcast process:

Each fog node broadcasts the information of the $Qw_j(t)$ to its neighboring fog nodes.

5.3. Performance Analysis

This subsection further discusses the performance of the DAOWA algorithm in terms of the average queuing length of a workload and the average TSD. Let $\boldsymbol{\pi}^*(t) = \left(\pi^{(1)*}(t), \ldots, \pi^{(j)*}(t), \ldots, \pi^{(R)*}(t)\right)$ be the optimal decision based on the DAOWA algorithm, $Xw_j^{A*}(t)$ and $E\left[T_{avg}(\pi^*(t))\right]$ denote the corresponding workload and average TSD for fog node j, which can be achieved by the S-only policy [33]. Then, we have:

$$\begin{gathered} \Delta L(t) + V \cdot E\left[T_{avg}(\boldsymbol{\pi}(t))\right] \leq \\ B + E\left[\sum_{j=1}^{R}\left(Qw_j(t) \cdot \left(Xw_j^{A*}(t) - P_j \cdot \phi\right)\right)\right] + V \cdot E\left[T_{avg}(\boldsymbol{\pi}^*(t))\right]. \end{gathered} \tag{37}$$

We assume that the workload of the fog node is finite. Thus, there exists a finite constant $C = \max\left(E\left[Xw_j^{A*}(t)\right]\right)$. Let $\overline{T^*_{avg}}(t) = E\left[T_{avg}(\boldsymbol{\pi}^*(t))\right]$, we have:

$$\Delta L(t) + V{\cdot}E[T(\boldsymbol{\pi}(t))] \le B + \left(C - P_j{\cdot}\phi\right){\cdot}E\left[\sum\nolimits_{j=1}^{R} Qw_j(t)\right] + V{\cdot}\overline{T^*_{avg}}(t). \tag{38}$$

Summing both sides of the above inequality over T slots and disregarding the negative quantities, we have:

$$\begin{aligned} &E[L(T)] - E[L(0)] + V{\cdot}\textstyle\sum_{t=0}^{T-1} E\left[T_{avg}(\boldsymbol{\pi}^*(t))\right] \\ &\le T{\cdot}B + C{\cdot}\textstyle\sum_{t=0}^{T-1} E\left[\sum_{j=1}^{R} Qw_j(t)\right] + V{\cdot}\sum_{t=0}^{T-1} \overline{T^*_{avg}}(t). \end{aligned} \tag{39}$$

Owing to $L(0) = 0$ and $\mathrm{E}[L(T)] \ge 0$, we arrange the terms in the above inequality by dividing $V{\cdot}T$ as follows:

$$\frac{1}{T}{\cdot}\sum\nolimits_{t=0}^{T-1} E\left[T_{avg}(\boldsymbol{\pi}(t))\right] \le \frac{B}{V} + \frac{1}{T}{\cdot}\sum\nolimits_{t=0}^{T-1} \overline{T^*_{avg}}(t) + \frac{C}{T}{\cdot}\sum\nolimits_{t=0}^{T-1} E\left[\sum\nolimits_{j=1}^{R} Qw_j(t)\right]. \tag{40}$$

Taking the limits as $T \to \infty$ and letting $\overline{T^*_{avg}} = \lim\limits_{T\to\infty} \frac{1}{T}{\cdot}\sum_{t=0}^{T-1} \overline{T^*_{avg}}(t)$, we obtain the inequality as:

$$\frac{1}{T}{\cdot}\sum\nolimits_{t=0}^{T-1} E\left[T_{avg}(\boldsymbol{\pi}(t))\right] \le \frac{B}{V} + \frac{1}{T}{\cdot}\sum\nolimits_{t=0}^{T-1} \overline{T^*_{avg}}(t) + \frac{C}{T}{\cdot}\sum\nolimits_{t=0}^{T-1} E\left[\sum\nolimits_{j=1}^{R} Qw_j(t)\right]. \tag{41}$$

Let $\overline{Qw^{max}} = \max\left(\lim\limits_{T\to\infty} \frac{1}{T}{\cdot}\sum_{t=0}^{T-1}\sum_{j=1}^{R} E\left[Qw_j(t)\right]\right)$, then:

$$\lim_{T\to\infty} \frac{1}{T}{\cdot}\sum\nolimits_{t=0}^{T-1} E[T(\boldsymbol{\pi}(t))] \le \frac{B}{V} + \overline{T^*_{avg}} + C{\cdot}\overline{Qw^{max}}. \tag{42}$$

Similarly, we assume that there exists a finite constant D such that $P_j{\cdot}\phi - Xw_j^{A*}(t) \le D$ by the S-only policy [33]. Let $\overline{T_{avg}} = \lim\limits_{T\to\infty} \frac{1}{T} \cdot \sum_{t=0}^{T-1} E\left[T_{avg}(\boldsymbol{\pi}(t))\right]$, in which case we obtain the following inequality by a similar process:

$$\lim_{T\to\infty} \frac{1}{T}{\cdot}\sum\nolimits_{t=0}^{T-1} E\left[\sum\nolimits_{j=1}^{R} Qw_j(t)\right] \le \frac{B + V{\cdot}\left[\overline{T^*_{avg}} - \overline{T_{avg}}\right]}{D}. \tag{43}$$

The bounds in Equations (42) and (43) indicate a $\left[O\left(\frac{1}{V}\right), O(V)\right]$ tradeoff between the average TSD and average queuing length of a workload. The average TSD approaches the DAOWA-generated $\overline{T^*_{avg}}$ when parameter V is sufficiently large, but ignores the stability of the average queue of a workload in the fog node. By contrast, the average queuing length of a workload approaches its optimum when the value of V is small. Tuning the parameter V can achieve the optimal objective of minimizing $\overline{T^*_{avg}}$ as well as guaranteeing the stability of queuing the workload of the fog nodes.

6. Performance Evaluation

6.1. Simulation Environment Settings

We choose CloudSim [34] as our simulation platform, and we extended CloudsSim by adding new settings to conduct our experiments, which are similar to those in previous reports [35,36]. The simulation scenario comprises three regions and a cloud center, with each region endowed with one fog node and a number of IoT end devices. Simulation parameters and the topology are listed in Table 1. Based on [37], we also use the Poisson distribution with the vector $\boldsymbol{\lambda}(t) = \left[\overline{\lambda_1(t)}, \overline{\lambda_2(t)}, \overline{\lambda_3(t)}\right]$ representing the expected number of arrivals to model the task generation rate of the end tier in the three regions. For each time slot, the length of the corresponding task instruction (million instructions, MI) follows an exponential distribution with an expected length vector $\boldsymbol{l}(t) = \left[\overline{l_1(t)}, \overline{l_2(t)}, \overline{l_3(t)}\right]$. Similarly,

the data size follows a uniform distribution with an average size vector $\boldsymbol{d}(t) = \left[\overline{d_1(t)}, \overline{d_2(t)}, \overline{d_3(t)}\right]$. We set the frequency of each central processing unit (CPU) cycle of VMs in the cloud center to 3.2 GHz, which is faster than that of each fog node (2.0 GHz). The *CPI* of both the cloud center and the fog node are 2.5. Based on the real situation, the f2f bandwidth is 54 Mbps, whereas the f2c bandwidth is 1 Gbps. The mean f2f and f2c propagation delays are 1 ms and 50 ms, respectively.

Table 1. The basic parameter settings.

Parameters	IoT Region 1	IoT Region 2	IoT Region 3
$F_j^{fog}(GHz)$	2.0	2.0	2.0
CPI	2.5	2.5	2.5
CPUs	1	1	1
$\overline{\lambda_j(t)}$	$0.5\overline{\lambda_2(t)}$	0.10	$0.5\overline{\lambda_2(t)}$
$\overline{l_j(t)}$(10MI)	$0.5\overline{l_2(t)}$	0.6	$0.5\overline{l_2(t)}$
$\overline{d_j(t)}$(Mbits)	U[1, 10]	U[1, 10]	U[1, 10]
Neighbor	Region 2	Region 1, 3	Region 2
F^{cloud}(GHz)		3.2	
CPI		2.5	
f2f bandwidth (Mbps)		54	
f2c bandwidth (Gbps)		1	
f2f propagation delay (ms)		1	
f2c propagation delay (ms)		50	

6.2. Impact of V on Average Task Service Delay

Figure 3 shows the impact of control parameter V, which is defined in Equation (29), on the average TSD. When $V = 0$, the proposed algorithm degrades into a workload-based Lyapunov workload allocation approach. According to the theoretical analysis, the policy should stabilize $Qw_j(t)$, and the best scheme should ensure that $Qw_j(t) = 0$. Thus, the optimal policy schedules all tasks for processing in the cloud center. However, owing to the large f2c propagation delay, it is unsurprising that the average TSD is the largest in comparison with those obtained when $V > 0$. When $0 < V < 1$, the average TSD first decreases as the value of V increases. When $1 < V < 10$, the average TSD reaches a small value. This is because our proposed algorithm attempts to find the optimal workload allocation and task scheduling policy based on the tradeoff of the penalty over the workload in the drift-plus-penalty formulation. The average TSD starts to increase when $V > 10$, because the larger V is, the lower the $Qw_j(t)$ affection is. Thus, it increases the queuing delay as $Qw_j(t)$ increases. The result also verifies the analysis in Equation (42), where longer $\overline{Qw^{max}}$ leads to a larger bound for the average TSD.

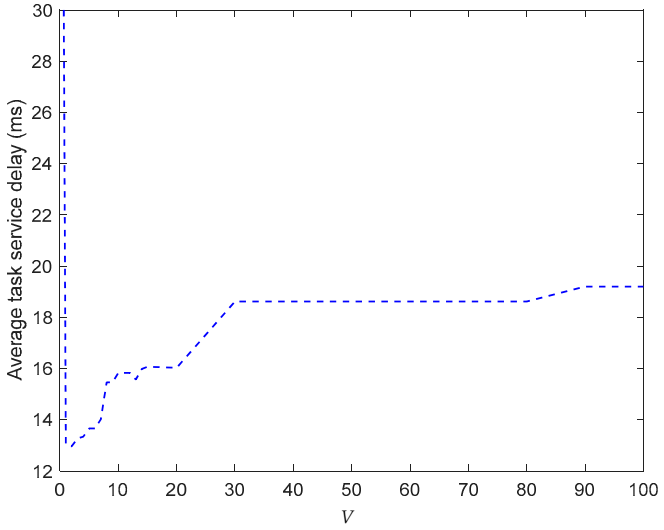

Figure 3. Average task service delay vs. *V*.

6.3. Comparison of the Task Service Delay of the Regions

We evaluated the efficiency of the proposed DAOWA algorithm by comparing with other algorithms, including fog-processing algorithm (Fog), cloud processing algorithm (Cloud), fog-to-cloud cooperation algorithm (F&C) and greedy algorithm (Greedy). In the Fog algorithm, all tasks are processed at the local fog node. In the Cloud algorithm, all tasks are offloaded to the cloud center. The F&C algorithm decides whether the task should be processed at the local fog node or offloaded to the cloud center with the aim of minimizing the delay. The Greedy algorithm only considers the TSD at the current time slot as the objective to determine the workload allocation decision. The average TSD of the three regions defined in Table 1 was computed by the aforementioned four algorithms and is plotted in Figure 4.

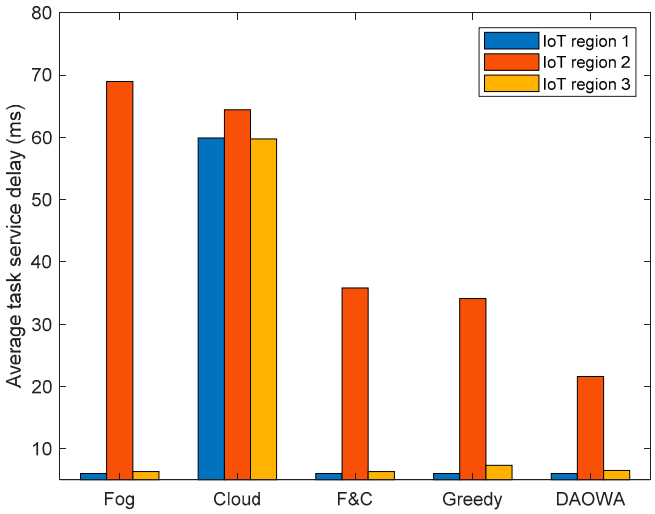

Figure 4. Average task service delay for each of the three IoT regions computed with the different algorithms.

With reference to Table 1, because the task arrival rate and task instruction length in IoT region 2 are larger than those in the other two IoT regions, the large workload arrival at IoT region 2 yields the largest average TSD. Because the tasks can only be processed at the local fog node when using the Fog algorithm, the limited computation capability of the fog node and the large workload arrival at IoT region 2 causes significant queuing delay to drastically increase the average TSD. Although the cloud center has sufficient computation capability to eliminate the queuing delay, the large f2c propagation delay increases the average TSD of each IoT region. Therefore, the results in Figure 4 demonstrate that it is vital to enable the fog node to cooperate with the cloud. The advantage is also demonstrated by the result of the F&C algorithm. Compared with the F&C algorithm, although the Greedy algorithm achieves a lower TSD, it ignores the workload stability of the fog node, which may cause a long queuing delay to limit the average TSD reduction. Our proposed algorithm (i.e., DAOWA) can improve the average TSD compared with the F&C algorithm. This is because the DAOWA algorithm not only enables the local fog node to cooperate with the cloud center, it also enables the local fog node to cooperate with its neighboring fog nodes. Furthermore, the tradeoff between the queuing delay and network delay is optimized by the DAOWA algorithm. Thus, the average TSD can be reduced significantly by the DAOWA algorithm.

6.4. Varying the Task Arrival Rate

We evaluated the efficiency of the proposed algorithm for different task arrival rates. Based on Table 1, we set $\overline{\lambda_1(t)} = \overline{\lambda_3(t)} = 0.5\overline{\lambda_2(t)}$.The value of V was set to 5. Then, we vary the task arrival rate of IoT region 2 from 0.02 to 0.14. The results are shown in Figure 5.

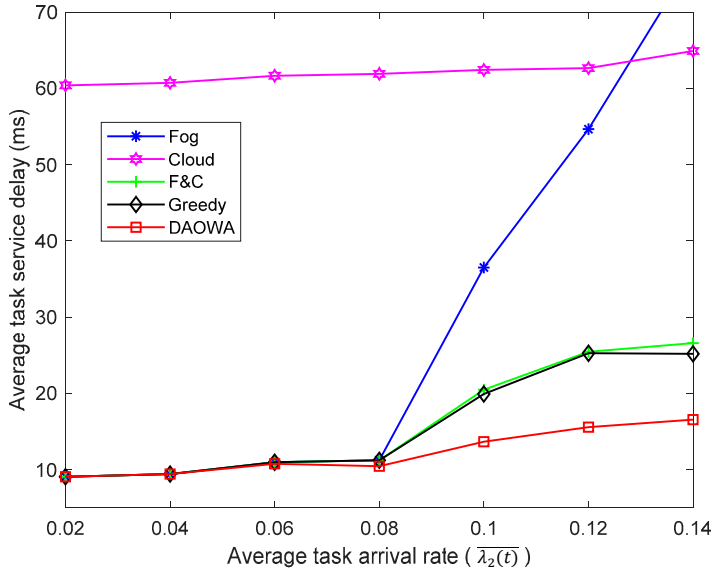

Figure 5. Average task service delay vs. task arrival rate.

When the task arrival rate is low ($\overline{\lambda_2(t)} < 0.08$), the computation capability of the local fog node is sufficient to guarantee the stability of $Qw_j(t)$. In this situation, the optimal policy enables the tasks to be processed at the local fog node rather than offloading them to other fog nodes or the cloud center. Thus, the F&C and Greedy algorithms allocate most of the workload to the local fog node, depending on the delay between the local fog node and the cloud center. The DAOWA algorithm also allocates most of the workload to the local fog node. Thus, all the algorithms achieve similar TSDs.

When the task arrival rate increases ($\overline{\lambda_2(t)} > 0.08$)), the Fog algorithm causes severe queuing delays owing to the limited computation capability of the fog node. Thus, the Fog algorithm exacerbates the average TSD. Furthermore, when the task arrival rate is sufficiently large (e.g., $\overline{\lambda_2(t)} > 0.12$), the average TSD obtained by the Fog algorithm could be larger than that obtained by the Cloud algorithm. Because the F&C algorithm enables the local fog node to cooperate with the cloud center, it succeeds in lowering the average TSD. However, the f2c propagation delay affects the performance of the F&C algorithm. The average TSD obtained with the F&C algorithm increases when the task arrival rate increases. The Greedy algorithm ignores the workload stability of the fog node, and a long queuing delay may impair the advantage of the fog-to-fog coordination. Thus, the performance of the Greedy algorithm is similar to that of the F&C algorithm. Compared with the other algorithms, the DAOWA algorithm is not only aware of the workload of the fog node, but also considers the network delay. It guarantees the workload stability of the fog node and avoids a large network delay. This means that the DAOWA algorithm is able to adaptively achieve the lowest average TSD according to the task arrival rate. In spite of this, the increasing task arrival rate results in more tasks being offloaded to the cloud center. In this sense, the large f2c propagation delay increases the average TSD.

Because of the large propagation delay, the average TSD obtained by the Cloud algorithm is larger than that obtained by the F&C, Greedy and DAOWA algorithms. Because of the large f2c bandwidth and the sufficient computation capability of the cloud center, the average TSD obtained with the Cloud algorithm changes slowly when using a one user-one VM mode.

6.5. Varying the Task Instruction Length

We also evaluate the efficiency of the Fog, Cloud, F&C, Greedy and DAOWA algorithms for various task instruction lengths. We adopt the parameter settings listed in Table 1 except for the average task instruction length. We set $\overline{l_1(t)} = \overline{l_3(t)} = 0.5\overline{l_2(t)}$, that is, the expected task length in IoT region 1 and IoT region 3 is half of that in IoT region 2. The value of V is set to 5. Then, we vary the expected task length of IoT region 2 from 0.2×10 MI to 1×10 MI. The results are shown in Figure 6.

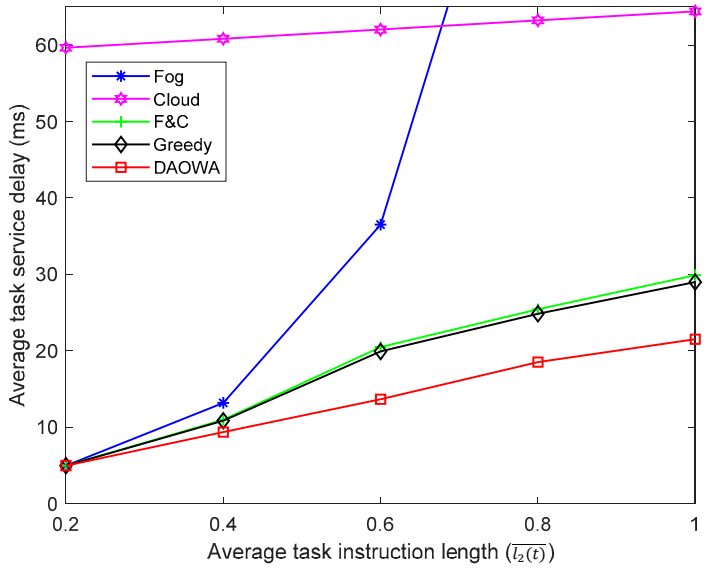

Figure 6. Average task service delay vs. task instruction length

When the task instruction length is short ($\overline{l_2(t)} < 0.4$), the computation capability of the local fog node can guarantee the workload stability for the low delay requirement. In this situation, most of the workloads do not need to be offloaded to the neighboring fog node and the cloud center. Thus, the performance of the Fog, F&C, Greedy and DAOWA algorithms is similar. However, owing to the large f2c propagation delay, the Cloud algorithm obtains the largest average TSD.

When the average task instruction increases ($\overline{l_2(t)} > 0.4$), the limited computation capability of the fog node is unable to stabilize Qw_j, whereupon the queuing delay increases markedly. Furthermore, a significant increase in the average task instruction length causes the queuing delay to exceed the f2c propagation delay, in which case the average TSD of the Fog algorithm becomes larger than that of the Cloud algorithm. Without fog-to-fog cooperation, the F&C algorithm offloads additional work to the cloud center, which increases the propagation and congestion delay, leading to a larger average TSD than that of the DAOWA algorithm. Similar with the experiment in Section 6.4, a long queuing delay under the Greedy algorithm may limit the advantage of the fog-to-fog coordination. Thus, the average TSD under the Greedy algorithm is similar to that under the F&C algorithm. Overall, regardless of the workload, the DAOWA algorithm always provides the lowest average TSD in comparison with other algorithms, as shown in Figure 6. This is because the DAOWA algorithm can dynamically switch among the local fog node, neighboring fog nodes, and the cloud center adaptively to guarantee the stability of the workload queue and minimize the TSD for various task instruction lengths.

6.6. Varying the Computing Speed of the Fog Node

To evaluate the efficiency of our proposed algorithm, we vary the frequency of the CPU cycle of the fog nodes from 1.0 GHz to 3.0 GHz according to the definition of computing speed in Equation (1). The other parameter settings are the same as those in Table 1. The results of this comparison are plotted in Figure 7.

Unsurprisingly, when the computing speed of the fog node is low ($F_j^{fog} < 2.0$ GHz), the poor computation capability of the fog node causes significant queuing delay. Consequently, the average TSD obtained by the Fog algorithm is enormous, even larger than the average TSD of the Cloud algorithm. The F&C algorithm enables the local fog node to cooperate with the cloud center to mitigate the poor performance of the local fog node. The DAOWA algorithm achieves the lowest average TSD by the help of neighboring fogs. Although the Greedy algorithm also enables the local fog node to cooperate with its neighboring fog nodes and the cloud center, it ignores the workload stability. Furthermore, because the heavy workload of the fog nodes under the Greedy algorithm will cause

a long queuing delay, the average TSD under the Greedy algorithm is larger than that under the DAOWA algorithm.

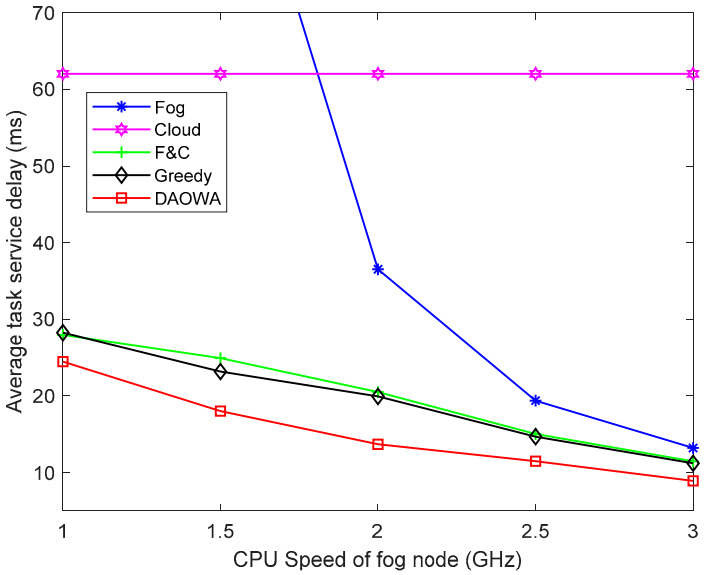

Figure 7. Average task service delay vs. fog node computing speed.

Increasing the computing speed of the fog node ($F_j^{fog} > 2.0$ GHz) leads to an increase in its performance, which reduces the average TSD when using the Fog, F&C, and DAOWA algorithms. Compared with other algorithms, the DAOWA algorithm can adaptively allocate the workload according to the computing speed of the fog node. The DAOWA algorithm stabilizes the workload and avoids large processing delays. Thus, the DAOWA algorithm obtains the lowest average TSD. Furthermore, when $F_j^{fog} = 3.0$ GHz, because of the considerable increase in the computing speed of the fog node, the DAOWA algorithm does not need to enable the local fog node to cooperate with other fog nodes. Thus, the result of the DAOWA algorithm is similar to that of the Fog, F&C and Greedy algorithms. In real situations, it would not be possible to significantly increase the computing speed of the fog node.

6.7. Varying the f2c Propagation Delay

We evaluated the efficiency of our proposed algorithm by varying the f2c propagation delay from 20 ms to 80 ms. The other parameter settings are the same as those in Table 1. The results are shown in Figure 8.

As shown in Figure 8, because the Fog algorithm cannot be affected by the f2c propagation delay, the average TSD of the Fog algorithm does not change. Furthermore, owing to the limited computation capability of the fog node, the average TSD calculated with the Fog algorithm is larger than that obtained by the F&C, Greedy and DAOWA algorithms. In addition, the average TSD of the Cloud algorithm increases when the f2c propagation delay increases. This demonstrates that the f2c propagation delay mainly affects the average TSD of the Cloud algorithm because of the long-distance data transmission between the fog node and cloud center. However, owing to the cooperation between the local fog node and cloud center, the average TSD of the F&C algorithm is less than that obtained by the Cloud and Fog algorithms.

When the f2c propagation delay is small (20 ms), the average TSD under the F&C and Greedy algorithms are lower than that of the Fog algorithm, and are close to that of the DAOWA algorithm. This is because the computation capability of the cloud center is sufficient, the DAOWA and Greedy algorithms do not need to allocate the workload to neighboring fog nodes when the f2c propagation delay is small. In this case the optimized scheme enables the local fog node to cooperate directly with

the cloud center. Another noteworthy observation is that the average TSD of the F&C, Greedy and DAOWA algorithms increases as the f2c propagation delay increases. When the f2c propagation delay increases, the Greedy algorithm will enable the local fog node to cooperate with its neighboring fog nodes to prevent the average TSD from increasing fast. Thus, the average TSD under the Greedy algorithm is lower than that under the F&C algorithm. The average TSD under the DAOWA algorithm is the lowest. This demonstrates the ability of the DAOWA algorithm to optimize the policy to allocate the workload according to the system status.

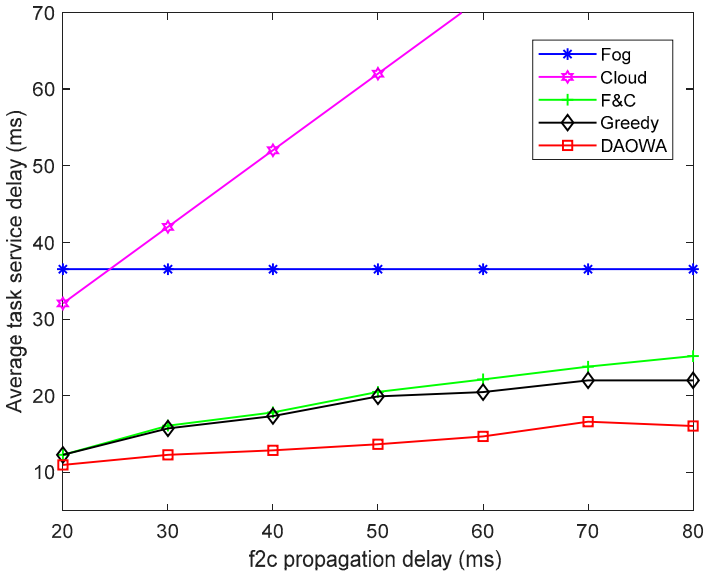

Figure 8. Average task service delay vs. f2c propagation delay.

7. Conclusions

This paper proposes a fog-to-fog cooperation scheme to minimize the task-processing delay in an IoT-fog-cloud system. Based on the topology framework of the system in real situations, we have built a systematic, comprehensive, and analytic time-varying queuing model. This model considers the computation capability, the amount of traffic, and the network transmission delay. In particular, we have formulated and developed a delay-aware workload allocation scheme, named DAOWA. We have analyzed the Lyapunov drift-plus-penalty properties of the time-varying queuing model of the fog tier and design the algorithm to minimize the drift-plus-penalty to reduce the average TSD. The theoretical analysis and simulation results demonstrate the ability of the proposed algorithm can minimize the task service delay efficiently.

Author Contributions: Conceptualization, L.L.; Data curation, L.L.; Formal analysis, L.L.; Investigation, L.L. and M.G.; Methodology, L.L. and M.G.; Project administration, Q.G.; Resources, H.M.; Software, L.L.; Supervision, L.M.; Validation, L.L.; Visualization, H.M.; Writing – original draft, L.L.; Writing – review and editing, M.G. and L.M.

Funding: This work was supported by NSFC project (No. 61901128, 61471173, 61671208, 6177119), and Project of Guangdong Province Science and Technology Program (No. 2017A010101027).

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

Appendix A

Proof of Theorem 1. By squaring both sides of Equation (5), we can obtain:

$$\left[Qw_j(t+1)\right]^2 = \left\{\max\left[Qw_j(t) + Xw_j^A(t) - P_j{\cdot}\phi, 0\right]\right\}^2.$$

Then,

$$\left[Qw_j(t+1)\right]^2 \le \left[Qw_j(t) + Xw_j^A(t) - P_j{\cdot}\phi\right]^2$$

We have

$$\left[Qw_j(t+1)\right]^2 \le \left\{\left[Qw_j(t)\right]^2 + 2\cdot Qw_j(t){\cdot}\left[Xw_j^A(t) - P_j{\cdot}\phi\right] + \left[Xw_j^A(t) - P_j{\cdot}\phi\right]^2\right\}.$$

According to the definition of Lyapunov drift in Equation (27), we have the following:

$$\begin{aligned}\Delta L(t) &= E\left[\tfrac{1}{2}\textstyle\sum_{j=1}^{R}\left[Qw_j(t+1)\right]^2 - \tfrac{1}{2}\sum_{j=1}^{R}\left[Qw_j(t)\right]^2 \middle| \boldsymbol{Qw}(t)\right] \\ &\le \tfrac{1}{2}E\left[\textstyle\sum_{j=1}^{R}\left(Xw_j^A(t) - P_j{\cdot}\phi\right)^2 \middle| \boldsymbol{Qw}(t)\right] + E\left[\sum_{j=1}^{R}\left(Qw_j(t){\cdot}\left(Xw_j^A(t) - P_j{\cdot}\phi\right)\right) \middle| \boldsymbol{Qw}(t)\right].\end{aligned}$$

Because the number of tasks and the length of task instructions in the fog system are finite, $Xw_j^A(t)$ has its upper bound ${Xw_j}^{max}(t)$, i.e., $Xw_j^A(t) \le {Xw_j}^{max}(t)$. Furthermore, the computing speed of the fog node can be considered to be a constant value. Thus,

$$\left[Xw_j^A(t) - P_j{\cdot}\phi\right]^2 \le max\left\{\left[{Xw_j}^{max}(t)\right]^2, \left[P_j{\cdot}\phi - Xw_j^A(t)\right]^2\right\}.$$

Let $B = \max\left\{\frac{1}{2}\left[{Xw_j}^{max}(t)\right]^2, \frac{1}{2}\left[P_j{\cdot}\phi - Xw_j^A(t)\right]^2\right\}$. Subsequently, we have

$$\Delta L(t) \le B + E\left[\sum\nolimits_{j=1}^{R}\left(Qw_j(t){\cdot}\left(Xw_j^A(t) - P_j{\cdot}\phi\right)\right) \middle| \boldsymbol{Qw}(t)\right],$$

which proves Theorem 1. □

References

1. Weiner, M.; Jorgovanovic, M.; Sahai, A.; Nikolié, B. Design of a low-latency, high-reliability wireless communication system for control applications. In Proceedings of the 2014 International conference on communications (ICC), Sydney, NSW, Australia, 10–14 June 2014; pp. 3829–3835.
2. Chiang, M.; Zhang, T. Fog, IoT: An overview of research opportunities. *IEEE Internet Things J.* **2015**, *3*, 854–864. [CrossRef]
3. Buyya, R.; Yeo, C.S.; Venugopal, S.; Broberg, J.; Brandic, I. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Gener. Comput. Syst.* **2009**, *25*, 599–616. [CrossRef]
4. Dustdar, S.; Guo, Y.; Satzger, B.; Truong, H.L. Principles of elastic processes. *IEEE Internet Comput.* **2011**, *15*, 66–71. [CrossRef]
5. Parwekar, P. From Internet of Things towards cloud of things. In Proceedings of the 2nd International Conference on Computer and Communication Technology (ICCCT-2011), Allahabad, India, 15–17 September 2011; pp. 329–333.
6. Dastjerdi, A.V.; Buyya, R. Fog computing: Helping the internet of things realize its potential. *Computer* **2016**, *49*, 112–116. [CrossRef]
7. Ning, Z.; Kong, X.; Xia, F.; Hou, W.; Wang, X. Green and sustainable cloud of things: Enabling collaborative edge computing. *IEEE Commun. Mag.* **2019**, *57*, 72–78. [CrossRef]
8. Masip-Bruin, X.; Tashakor, G.; Jukan, A.; Ren, G.J. Foggy clouds and cloudy fogs: A real need for coordinated management of fog-to-cloud computing systems. *IEEE Wireless Commun.* **2016**, *23*, 120–128. [CrossRef]
9. Bonomi, F.; Milito, R.; Zhu, J.; Addepalli, S. Fog computing and its role in the Internet of Things. In Proceedings of the 1st Edition of the MCC Workshop on Mobile Cloud Computing, Helsinki, Finland, 17 August 2012; pp. 13–16.

10. Chen, X.; Jiao, L.; Li, W.; Fu, X. Efficient multi-user computation offloading for mobile-edge cloud computing. *IEEE/ACM Trans. Netw.* **2016**, *24*, 2795–2808. [CrossRef]
11. Yu, W.; Liang, F.; He, X.; Hatcher, W.; Lu, C.; Yang, J.L.X. A survey on the edge computing for the Internet of Things. *IEEE Access* **2018**, *6*, 6901–6919. [CrossRef]
12. da Silva, R.A.; da Fonseca, N.L. On the Location of Fog Nodes in Fog-Cloud Infrastructures. *Sensors* **2019**, *19*, 2445. [CrossRef]
13. Neely, M.J. Stochastic network optimization with application to communication and queueing systems. In *Synthesis Lectures on Communication Networks*; Morgan & Claypool Press: San Rafael, CA, USA, 2010.
14. Tassiulas, L.; Ephremides, A. Stability Properties of Constrained Queueing Systems and Scheduling Policies for Maximum Throughput in Multihop Radio Networks. *IEEE Trans. Autom. Contr.* **1992**, *37*, 1936–1948. [CrossRef]
15. Xu, J.; Chen, L.; Ren, S. Online Learning for Offloading and Autoscaling in Energy Harvesting Mobile Edge Computing. *IEEE Trans. Cogn. Commun. Netw.* **2017**, *3*, 361–373.
16. Battula, S.K.; Garg, S.; Naha, R.K.; Thulasiraman, P.; Thulasiram, R. A Micro-Level Compensation-Based Cost Model for Resource Allocation in a Fog Environment. *Sensors* **2019**, *19*, 2954.
17. Amoretti, M.; Grazioli, A.; Zanichelli, F. A Modeling and Simulation Framework for Mobile Cloud Computing. *Simul. Model. Pract. Theory* **2015**, *58*, 140–156. [CrossRef]
18. Fan, Q.; Ansari, N. Towards Workload Balancing in Fog Computing Empowered IoT. *IEEE Trans. Netw. Sci. Eng.* **2018**. [CrossRef]
19. Lyu, X.; Tian, H.; Jiang, L.; Vinel, A.; Maharjan, S.; Gjessing, S.; Zhang, Y. Selective Offloading in Mobile Edge Computing for the Green Internet of Things. *IEEE Netw.* **2018**, *32*, 54–60. [CrossRef]
20. Chang, Z.; Zhou, Z.; Ristaniemi, T.; Niu, Z. Energy Efficient Optimization for Computation Offloading in Fog Computing System. In Proceedings of the GLOBECOM 2017 IEEE Global Communications Conference, Singapore, Singapore, 4–8 December 2017; pp. 1–6.
21. Guo, S.; Liu, J.; Yang, Y.; Xiao, B.; Li, Z. Energy-Efficient Dynamic Computation Offloading and Cooperative Task Scheduling in Mobile Cloud Computing. *IEEE Trans. Mob. Comput.* **2019**, *18*, 319–333. [CrossRef]
22. Rahbari, D.; Nickray, M. Task offloading in mobile fog computing by classification and regression tree. *Peer Peer Netw. Appl.* **2019**, 1–19. [CrossRef]
23. Li, W.; Santos, I.; Delicato, F.C.; Pires, P.F.; Pirmez, L.; Wei, W.; Song, H.; Zomaya, A.; Khane, S. System modelling and performance evaluation of a three-tier Cloud of Things. *Future Gener. Comput. Syst.* **2017**, *25*, 599–616.
24. Wu, H.; Sun, Y.; Wolter, K. Energy-efficient decision making for mobile cloud offloading. *IEEE Trans. Cloud Comput.* **2018**. [CrossRef]
25. Ma, K.; Bagula, A.; Nyirenda, C.; Ajayi, O. An IoT-Based Fog Computing Model. *Sensors* **2019**, *19*, 2783. [CrossRef]
26. Yousefpour, A.; Ishigaki, G.; Gour, R.; Jue, J. On Reducing IoT Service Delay via Fog Offloading. *IEEE Internet Things J.* **2018**, *5*, 998–1010. [CrossRef]
27. Nan, Y.; Li, W.; Bao, W.; Delicato, F.; Pires, P.; Dou, Y.; Albert, Y. Adaptive energy-aware computation offloading for cloud of things systems. *IEEE Access* **2017**, *5*, 23947–23957. [CrossRef]
28. Deng, R.; Lai, R.L.; Luan, T.; Liang, H. Optimal Workload Allocation in Fog-Cloud Computing Towards Balanced Delay and Power Consumption. *IEEE Internet Things J.* **2016**, *3*, 1171–1181. [CrossRef]
29. Nawrocki, P.; Reszelewski, W. Resource usage optimization in Mobile Cloud Computing Computer Communications. *Comput. Commun.* **2017**, *99*, 1–12. [CrossRef]
30. Martonosi, M.; Brooks, D.; Bose, P. Modeling and analyzing CPU power and performance: Metrics methods and abstractions. In Proceedings of the SIGMETRICS 2001/Performance 2001-Tutorials, Cambridge, MA, USA, 16–20 June 2001; Available online: http://www.princeton.edu/~{}mrm/tutorial/hpca2001_tutorial.pdf (accessed on 21 June 2019).
31. Liu, F.; Zhou, Z.; Jin, H.; Li, B.; Li, B.; Jiang, H. On arbitrating the power-performance tradeoff in SaaS clouds. *IEEE Trans. Parallel Distrib. Syst.* **2014**, *25*, 2648–2658. [CrossRef]
32. Hua, P.; Dhelima, S.; Ning, H.; Qiu, T. Survey on fog computing: Architecture, key technologies, applications and open issues. *J. Netw. Comput. Appl.* **2017**, *98*, 27–42. [CrossRef]

33. Niu, Y.; Luo, B.; Liu, F.; Liu, J.; Li, B. When hybrid cloud meets flash crowd: Towards cost-effective service provisioning. In Proceedings of the 2015 IEEE Conference on Computer Communications (INFOCOM), Hong Kong, China, 26 April–1 May 2015; pp. 1044–1052.
34. Calheiros, R.N.; Ranjan, R.; Beloglazov, A.; Rose, C.A.F.D.; Buyya, R. CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw. Pract. Exp.* **2011**, *41*, 23–50. [CrossRef]
35. Zhua, X.; Guo, H.; Zhu, J.; Qin, X.; Wu, J. Towards Energy-Efficient Scheduling for Real-Time Tasks Under Uncertain Cloud Computing Environment. *J. Syst. Softw.* **2015**, *99*, 20–35.
36. Beloglazov, A.; Abawajy, J.; Buyya, R. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future Gener. Comput. Syst.* **2012**, *28*, 755–768. [CrossRef]
37. Calzarossa, M.C.; Vedova, M.L.D.; Massari, L.; Petcu, D.; Tabash, M.I.M.; Tessera, D. Workloads in the Clouds. In *Principles of Performance and Reliability Modeling and Evaluation*; Springer: Cham, Switzerland, 2016.