Joint Computation Offloading and Scheduling Optimization of IoT Applications in Fog Networks

Abhishek Hazra, Mainak Adhikari, Tarachand Amgoth, and Satish Narayana Srirama, *Senior Member, IEEE*

Abstract—In recent times, fog computing becomes an emerging technology that can exhilarate the cloud services towards the network edge for increasing the speeds up of various Internet-of-Things (IoT) applications. In this context, integrating priority-aware scheduling and data offloading allow the service providers to efficiently handle a large number of real-time IoT applications and enhance the capability of the fog networks. But the energy consumption has become skyrocketing, and it gravely affects the performance of the fog networks. To address this issue, in this paper, we introduce an Energy-Efficient Task Offloading (EETO) policy combined with a hierarchical fog network for handling energy-performance trade-off by jointly scheduling and offloading the real-time IoT applications.To achieve this objective, we formulate a heuristic technique for assigning a priority on each incoming task and formulate a stochastic-aware data offloading issue with an efficient virtual queue stability approach, namely the Lyapunov optimization technique. The proposed technique utilizes the current state information for minimizing the queue waiting time and overall energy consumption while meeting *drift-plus-penalty*. Furthermore, a constraint restricted progressive online task offloading policy is incurred to mitigate the backlog tasks of the queues. Extensive simulation with various Quality-of-Service (QoS) parameters signifies that the proposed EETO mechanism performs better and saves about 23.79% of the energy usage as compared to the existing ones.

Index Terms—IoT, fog networks, priority-aware scheduling, energy-efficiency offloading, Lyapunov optimization, Quality-of-Service.

1 Introduction

The rapid development of the Internet-of-Things (IoT) and embedded devices generate an enormous number of real-time applications for processing and establishing a smooth relationship between countless cool applications including smart home, smart traffic management, smart health-care etc. [1]. In this context, more and more real-time applications, including video streaming, gaming, movie, and human-machine interaction, etc. are emerged and attract great attention. As a result, the necessity of assigning priority of each real-time task as per its importance and schedule the tasks on the suitable computing devices by meeting different Quality-of-Service (QoS) objectives has become a serious challenge. In general, the end devices are typically resource-constrained, which have limited energy and processing capacity [2]. As a result, it is significant to achieve an energy-efficient and priority-aware task offloading strategy for further processing the real-time IoT applications on the remote computing devices.

Within the last few decades, many researchers focus on resource-rich centralized cloud servers for processing an enormous amount of real-time applications [3]. However, due to the unreliable connectivity, high propagation delay, and geo-distributed deployment, the centralized cloud servers fail to meet the low latency and energy demand criteria of the priority-aware tasks. As an extension, for improving the performance of the overall network, it is more efficient to move the priority-aware tasks to the edge of the networks. To address those challenges, CISCO has introduced a new computing paradigm namely fog computing [4] that leverage a multitude of collaborative end devices and local fog devices in terms of a base station, smart phone, laptop, etc. [5]. As a result, it is crushing to benefit from the fog computing specialties to minimize energy consumption and latency while distributing the intense workloads of the end devices through offloading.

According to the energy consumption formulation, the required energy consumption for processing the real-time applications on the remote computing devices should be the accumulation of data transmission time, transmission power, and processing capacity of the computing devices. Further, transmission power combines state of the channel such as available bandwidth, and data transmission rate. Thus, a higher transmission rate can be accomplished by extending the transmission power, which also decreases the transmission time. Thereby, the energy emission rate can be controlled by balancing the trade-off between the transmission power and transmission time. An optimal task scheduling strategy should always control this trade-off and increases the transmission rate when the channel is ideal and reduces when the workload is high.

1.1 Motivation

The challenges mentioned above can be handled efficiently with an elegant and standard Lyapunov optimization method and Lyapunov drift theory. This method is introduced to enable the constrained optimization of time averages in a general stochastic system without estimating

Corresponding author: Tarachand Amgoth.
A. Hazra and T. Amgoth are with the Indian Institute of Technology (Indian School of Mines) Dhanbad, Jharkhand, India. (e-mail: abhishekhazra.18dr0018@cse.iitism.ac.in, tarachand@iitism.ac.in).
M. Adhikari is with the Mobile & Cloud Lab, Institute of Computer Science, University of Tartu, Estonia (e-mail: mainak.ism@gmail.com).
S. N. Srirama is with the School of Computer and Information Sciences, University of Hyderabad, India (e-mail: satish.srirama@uohyd.ac.in).

the condition of the stochastic system in future slots. The technique first observes the constraints of the system and constructs virtual queues with the objective of optimizing the time average constraints, ensuring the stability of the queues. Further, to optimize the objective function in each time slot, a drift plus weighted penalty function is introduced. Intuitively, the weighted penalty controls the trade-off between the penalty optimization and backlog queue reduction. Motivated from the stability function of Lyapunov optimization method, we first construct multiple virtual queues for the incoming tasks with a heuristic priority assignment policy. Then, the updated Lyapunov optimization method has been introduced to seek the stability of the queues and a utility function is designed for jointly minimizing the energy constraint and queue stability with the *drift-plus-penalty* framework.

1.2 Contributions

The standard Lyapunov optimization function is guaranteed to stability of the virtual queues. However, this strategy cannot ensure the energy constraint trade-off for transmitting and processing the real-time tasks on the remote computing devices in fog networks. Therefore, the approach must be modified to conform the adaptability of this situation. Further, we add a constrained restriction mechanism for offloading the scheduled tasks by ensuring to meet the energy constraint of the fog networks. The key contributions of the proposed Energy-Efficient Task Offloading (EETO) policy are described as follows.

- We develop a priority-aware queueing assignment policy that accounts for specifying a fixed priority on each incoming task based on the arrival frequency and execution deadline, and assign them on the suitable priority queues (virtual queues).
- By casting the formulated objective function into a Lyapunov optimization framework, we design an efficient virtual queue stability approach to obtain an effective scheduling and offloading decision policy for the priority-aware tasks and utilizes the current system information in each time frame.
- We design a constrained restricted online task offloading policy to efficiently offload the scheduled tasks to the suitable remote computing devices with lower computational complexity.
- Finally, we evaluate the proposed strategy through extensive simulation runs over different baseline algorithms. Superior results demonstrate that the proposed strategy achieves better performance over various QoS parameters, including queue waiting time, task offloading delay, energy usage, and throughput.

The rest of the paper is structured as follows. Section 2 discusses various state-of-the-art algorithms for real-time IoT applications along with their merits. Section 3 presents the system model followed by the problem formulation of the work. The proposed EETO strategy is examined in Section 4. The empirical evaluation of the proposed EETO strategy is discussed in Section 5. Finally, the conclusion and future research direction are presented in Section 6.

2 Related Work

In recent times, several approaches have been developed for making an efficient task offloading strategy in fog network. Lingjun *et al.* have proposed a Device-to-Device (D2D) task offloading policy for making an energy-efficient and incentive-assisted fog environment [6]. Jie *et al.* [7] have introduced a combined Lyapunov optimization and Gibbs sampling-based task offloading technique in mobile edge environment. Jiao *et al.* [8] have proposed a mixed-integer nonlinear energy-acquainted task offloading policy for reducing the energy consumption of the network. Lixing *et al.* [9] have examined an energy-effective mobile edge offloading strategy for maximizing the system performance and efficiency. Lin *et al.* [10] have investigated an energy-efficient task scheduling and offloading strategy for minimizing cost of the network. An energy-aware task offloading strategy has been introduced by Yang *et al.* [11] in a homogeneous fog environment to find an optimal scheduling sequence of the tasks. Yiming *et al.* [12] have addressed a hierarchical computation offloading scheme for reducing the total energy consumption cost in a non-orthogonal fog network.

Queueing theory is a well-known and widely adopted technique for controlling the data-flow of the network. Nowadays, for minimizing the energy utilization and latency, the researchers prefer to combine queueing theory and Lyapunov optimization technique in a complex environment such as fog network [13]. For example, Tomoya *et al.* [14] have proposed a queuing network for assigning inter-arrival tasks in a fog network to minimize the average waiting time of the network. Su *et al.* [15] have designed a cloud-based energy optimal application offloading policy by modifying the Lyapunov technique. In [16], Wang *et al.* have developed a VariedLen algorithm to reduce power-performance trade-offs in the cloud radio access network. Mukherjee *et al.* [17], have designed a deadline aware fair scheduling strategy for distributing workload in the inter fog network. Lei *et al.* [18] have proposed a resource allocation and real-time offloading strategy in a heterogeneous fog queuing environment. A hierarchical workload allocation scheme has been designed by Qiang *et al.* [19] in the edge network for reducing the total processing time. Jitender *et al.* [20] have analysed a QoS-aware computation offloading policy for reducing resource utilization cost in a heterogeneous mobile-cloud environment. Liqing *et al.* [21] have investigated an energy-aware strategy for reducing the processing time and latency in a fog network. Yucen *et al.* [22] have developed a energy-aware data offloading strategy for minimizing the energy and cost of the network.

There are two significant difficulties for designing an energy-efficient task offloading strategy in fog networks. Firstly, ***how to develop an energy-efficient task scheduling policy for priority-aware tasks***. This helps the system to determine task importance and guaranteed to process them with minimal delay. Secondly, ***how to predict the availability of computing resources prior, in-order to obtain a low complexity offloading strategy***. This helps the system to obtain an efficient offloading decision using the current resources in each time stamp. This also allows the devices to utilize the total energy usage of the tasks while

meeting several QoS constraints. The majority of the task offloading strategies have not classified the tasks into multiple priorities (*i.e., delay-sensitive* and *computation-intensive*) based on their importance, even though it is one of the significant challenges in the IoT domain. Different from the above-mentioned works, in this paper, we jointly consider the priority-aware task classification and energy-aware task offloading strategy with an optimal scheduling policy in fog networks. A comparative analysis with five key design attributes between the existing frameworks and proposed work is bestowed in Table 1.

TABLE 1
Comparative Study Of The Existing Task Offloading Strategies

Existing Works	Task priority	Utilization of current resources	Low complexity framework	Priority-aware scheduling	Dynamic offloading
[23]	×	✓	✓	×	×
[16]	×	✓	×	×	×
[6]	×	✓	×	×	✓
[24]	×	×	✓	×	×
[17]	✓	✓	✓	×	×
[25]	✓	×	×	×	×
[22]	×	✓	✓	×	×
[26]	×	✓	✓	×	×
[27]	×	×	✓	×	✓
[28]	✓	×	×	×	×
Our work	✓	✓	✓	✓	✓

3 System Model and Problem Formulation

In this section, we first discuss the system model of the hierarchical fog network followed by the problem formulation of the proposed strategy. The important notations of the paper are referred to Table 2.

3.1 System Model

A typical hierarchical fog network is shown in Fig. 1. As exemplified in Fig. 1, let us consider that $\mathcal{F}$ be the set of fog devices, denoted as $\mathcal{F} = \{F_1, F_2, \ldots, F_o\}$ and $\mathcal{S}$ be a set of heterogeneous cloud servers, represented as $\mathcal{S} = \{S_1, S_2, \ldots, S_p\}$. In this model, $\mathcal{I}$ denotes the set of end devices, denoted as $\mathcal{I} = \{I_1, I_2, \ldots, I_m\}$. Here, we assume that the CPU capacity and energy consumption of an end device, *i.e.* f_{end}^{CPU} and E_{end}^{CPU} respectively, should be less than any fog device (f_{fog}^{CPU}, E_{fog}^{CPU}) and cloud servers (f_{cloud}^{CPU}, E_{cloud}^{CPU}). Similarly, the CPU capacity and energy consumption of a fog device is less than any cloud servers. Thus, we can assume that $f_{end}^{CPU} < f_{fog}^{CPU} < f_{cloud}^{CPU}$ and $E_{end}^{CPU} < E_{fog}^{CPU} < E_{cloud}^{CPU}$. Each end device can generate $\mathcal{T}$ number of *delay-sensitive* or *computation-intensive* tasks. The set of tasks are represented as $\mathcal{T} = \{T_1, T_2, \ldots, T_q\}$. The tasks can process either on local end devices or offload to the remote computing devices such as fog nodes or cloud servers through a set of local gateways $\mathcal{G}$, represented as $\mathcal{G} = \{G_1, G_2, \ldots, G_n\}$. The gateways are responsible to assign a priority on each incoming task according to its importance and and take an energy-efficient offloading decision with the updated Lyapunov optimization framework.

Here, we assume a binary offloading scenario, where a real-time task is processed on the local end device or offloaded to a remote computing device for processing fully. More importantly, a highly integrated or relatively simple task can not be partitioned and has to process as a whole on a single computing devices. Let $\mathcal{FS}$ be the set of remote computing devices including local fog devices and remote cloud servers, *i.e*, $\mathcal{FS} = (\mathcal{F} \cup \mathcal{S})$. The *delay-sensitive* tasks, referred to T_i^D, are offloaded to the fog devices and *computation-intensive* tasks, referred to T_i^C, are offloaded to the centralized cloud servers. In this model, each request generates from an end device with input or output data size *(in bits)*, denoted as T_i^{in} or T_i^{out}, respectively. Let $\mathcal{X}(i,j)$ referred to the assignment of ith task to the jth computing device, $\forall j \in (\mathcal{I} \cup \mathcal{FS})$. Then, $\mathcal{X}(i,j)$ is derived as follows.

$$\mathcal{X}(i,j) = \begin{cases} 1 & \text{if the } i\text{th task is assigned to the } j\text{th} \\ & \text{computing device: } \forall j \in (\mathcal{I} \cup \mathcal{FS}) \\ 0 & \text{otherwise.} \end{cases}$$

We also assume that each gateway device G_i offloads the tasks to the remote computing devices ($\mathcal{FS}$) in the network based on their availability and workloads.

TABLE 2
Important Notations

Symbols	Definition
$\mathcal{T}$	Total number of tasks
$\mathcal{S}$	Total number of cloud servers
$\mathcal{F}$	Total number of fog devices
$\mathcal{G}$	Total number of gateways
$\mathcal{I}$	Total number of end devices
λ_{jk}^D	Task arrival rate at the *delay-sensitive* queue
λ_{jk}^C	Task arrival rate at the *computation-intensive* queue
$\mathcal{FS}$	Total number of computing devices in the network
R_{ik}^{up}	Task uploading rate from ith IoT to kth gateway
B_{ij}^{in}	Uploading Bandwidth between ith and jth device
B_{ji}^{out}	Downloading Bandwidth between jth and ith device
γ_{kj}^D	*Delay-sensitive* task offloading probability
γ_{kj}^C	*Computation intensive* task offloading probability
T_{ij}^{up}	Uploading time between ith and jth device
T_{ji}^{down}	Downloading time between ith and jth device
f_j^{CPU}	CPU frequency of the jth device
E_{ij}^{pro}	Energy consumption at the jth computing device
E_{ij}^{total}	Total energy consumption to process ith task on jth computing device

3.1.1 Queueing Model

Here, we consider a time-slotted system, indexed by $T = \{t_1, t_2, \ldots, \}$, where the length of each time slot is Δt *(in sec)*. Further, we consider that the arrival rate of the tasks can be realistically modeled as a Poisson process with the density function $f(x) = \lambda_i e^{-\lambda_i x}$, where λ_i represents the task arrival rate from an end device. Let, α_{jk}, be the task offloading probability from jth end device to the local gateway. Thus, the task arrival rate at the local queue of kth gateway for taking further offloading decision is defined as $\lambda_{jk}^{rem} = \alpha_{jk} \times \lambda_i, \forall j \in \mathcal{I}$. Subsequently, the remaining tasks are processed on the local end devices. The arrival rate of the ith task that can process locally at the jth end devices is defined as $\lambda_j^{local} = (1 - \alpha_{jk}) \times \lambda_i$.

Let us consider that β_{jk} denotes the set of task comes under *delay-sensitive* queue of kth gateway for further offloading. Thus, the task arrival rate at the *delay-sensitive* queue is represented as $\lambda_{jk}^D = \beta_{jk} \times \lambda_{jk}^{rem}$. Similarly, the remaining tasks arrival rate at the *computation-intensive* queue of kth gateway is defined as $\lambda_{jk}^C = (1 - \beta_{jk}) \times \lambda_{jk}^{rem}$. Further, we

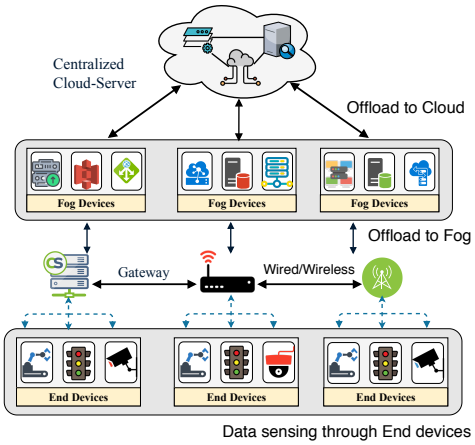

Fig. 1. Hierarchical fog networks model

denote that γ_{kj}^{D} and γ_{kj}^{C} represent the probabilities of *delay-sensitive* and *computation-intensive* task, respectively that are offloaded to the jth remote computing device ($\forall j \in \mathcal{F} \cup \mathcal{S}$) for further processing. Therefore, the task arrival rate from the kth gateway device to the jth fog device, $\forall j \in \mathcal{F}$ is defined as follows.

$$\begin{aligned}\lambda_i^{fog} &= \gamma_{kj}^{D} \times \lambda_{jk}^{D} + \gamma_{kj}^{C} \times \lambda_{jk}^{C} \\ &= \gamma_{kj}^{D} \times \beta_{jk} \times \lambda_{jk}^{rem} + \gamma_{kj}^{C} \times (1-\beta_{jk}) \times \lambda_{jk}^{rem}\end{aligned} \tag{1}$$

Likewise, the task arrival rate from the kth gateway device to the jth cloud server,$\forall j \in \mathcal{S}$ is expressed as follows.

$$\begin{aligned}\lambda_i^{cloud} &= (1-\gamma_{kj}^{D}) \times \lambda_{jk}^{D} + (1-\gamma_{kj}^{C}) \times \lambda_{jk}^{C} \\ &= (1-\gamma_{kj}^{D})\beta_{jk}\lambda_{jk}^{rem} + (1-\gamma_{kj}^{C})(1-\beta_{jk})\lambda_{jk}^{rem}\end{aligned} \tag{2}$$

3.1.2 Energy Consumption Model

To capture the key features of the energy consumption during computation and communication in fog networks, the energy consumption models of different modes are studied in this section. Specifically, we focus on energy consumption during task offloading and processing in local end devices and remote computing devices.

1. Local processing Mode: The incoming tasks from the set $\mathcal{T}$, which require minimum CPU frequency (f_i^{CPU}) for processing, *i.e.,* if the required CPU frequency of the ith task is less than the available CPU frequency of the local end device j, (*i.e.,* $f_i^{CPU} \leq f_j^{CPU}$), then the ith task is processed locally, where $\forall i \in \mathcal{T}$ and $\forall j \in \mathcal{I}$. As a result, the time to process the ith task on jth end device with probability $(1-\alpha_{ik})$ is defined as $P_{ij} = (1-\alpha_{jk}) \times \mathcal{X}(i,j)T_i^{in}/f_j^{CPU}$.

The local processing of a real-time task depends on the CPU frequency of the local end device instead of the communication delay. To simplify the analysis, we consider the energy consumption for running one bit of a task i at local end device, *i.e.* E_j^{CPU} (in Joules). Thus, the total energy consumption by running a task i at local end device j is defined as follows.

$$\begin{aligned}E_{ij}^{pro} &= P_{ij} \times E_j^{CPU} \qquad : \forall i \in \mathcal{T}, \forall j \in \mathcal{I} \\ &= (1-\alpha_{jk}) \times \frac{\mathcal{X}(i,j) \times T_i^{in}}{f_j^{CPU}} \times E_j^{CPU}\end{aligned} \tag{3}$$

2. Computation offloading Mode: Due to the resource-constrained processing capability, the end devices offload the large amount of tasks $\mathcal{T}$ to the resource-rich remote computing devices. Thus, the total offloading time of a task i depends on its uploading time, downloading time and processing time. Denote, hp_i be the channel power gain for offloading the ith task, $\forall T_i \in (T_i^D, T_i^C)$ in jth computing device, where, $\forall i \in \mathcal{T}$, $j \in (\mathcal{F}, \mathcal{S})$. Thus, the uploading rate (R_{ij}^{up}) of a task i on the jth computing device is defined as $R_{ij}^{\text{up}} = B_{ij}^{in} \log_2\left(1+\frac{\mathbb{P}_j^{\text{up}} hp_i}{\xi_i^2}\right)$, where B_{ij}^{in} signifies the obtainable bandwidth utilization between ith end device and jth computing device. $\mathbb{P}_j^{up}$ denotes the required transmission power to offload a task on jth computing device and ξ_i^2 is a constant additive white Gaussian noise of end device. The transmission time to offload a task to a nearby computing device can be expressed as follows.

$$T_{ij}^{\text{up}} = \frac{\mathcal{X}(i,j) \times T_i^{in}}{R_{ij}^{\text{up}}} \qquad : \forall i \in \mathcal{T}, j \in (\mathcal{F}, \mathcal{S}) \tag{4}$$

Consequently, the corresponding uploading energy consumption of task i while transmitting to the remote computing device j is defined as follows.

$$\begin{aligned}E_{ij}^{\text{up}} &= T_{ij}^{\text{up}} \times \mathbb{P}_j^{\text{up}} \qquad : \forall i \in \mathcal{T}, j \in (\mathcal{F}, \mathcal{S}) \\ &= \frac{\mathcal{X}(i,j) \times T_i^{in} \times \mathbb{P}_j^{up}}{B_{ij}^{in} \times log_2\left(1+\frac{\mathbb{P}_j^{up} \times hp_i}{\xi_i^2}\right)}\end{aligned} \tag{5}$$

Where, $\forall i \in \mathcal{T}$, $j \in (\mathcal{F}, \mathcal{S})$. Now, the total processing time to execute the ith task $\left[\forall i \in (T_i^D, T_i^C)\right]$ on the jth computing device $\forall j \in (\mathcal{F}, \mathcal{S})$ is given as follows.

$$P_{ij} = \begin{cases} \gamma_{kj}^{D} \times \frac{\mathcal{X}(i,j)T_i^{in}}{f_j^{CPU}}, & \text{if } T_i \in T_i^D \ \& \ j \in \mathcal{F} \\ \gamma_{kj}^{C} \times \frac{\mathcal{X}(i,j)T_i^{in}}{f_j^{CPU}}, & \text{if } T_i \in T_i^C \ \& \ j \in \mathcal{F} \\ (1-\gamma_{kj}^{D}) \times \frac{\mathcal{X}(i,j)T_i^{in}}{f_j^{CPU}}, & \text{if } T_i \in T_i^D \ \& \ j \in \mathcal{S} \\ (1-\gamma_{kj}^{C}) \times \frac{\mathcal{X}(i,j)T_i^{in}}{f_j^{CPU}}, & \text{if } T_i \in T_i^C \ \& \ j \in \mathcal{S} \end{cases} \tag{6}$$

Where, f_j^{CPU} denotes the CPU frequency of the jth computing device. The task arrival rate on remote fog devices and cloud servers are represented as λ_i^{fog} and λ_i^{cloud}, respectively. Further, the service rate of the remote computing devices is defined as $f_j^{CPU}/\mathcal{X}(i,j)T_i^{in}, \forall i \in \mathcal{T}, j \in (\mathcal{F}, \mathcal{S})$. According to $M/M/1$ queueing model [29], the queue waiting time in jth computing device is defined as follows.

$$\mathcal{Q}_{ij} = \begin{cases} \frac{\lambda_i^{fog}\left(\mathcal{X}(i,j)T_i^{in}\right)^2}{f_j^{CPU}\left(f_j^{CPU} - \lambda_i^{fog}\left(\mathcal{X}(i,j)T_i^{in}\right)\right)}, & \text{if } j \in \mathcal{F} \\ \frac{\lambda_i^{cloud}\left(\mathcal{X}(i,j)T_i^{in}\right)^2}{f_j^{CPU}\left(f_j^{CPU} - \lambda_i^{cloud}\left(\mathcal{X}(i,j)T_i^{in}\right)\right)}, & \text{if } j \in \mathcal{S} \end{cases} \tag{7}$$

The total queueing delay for remote execution is $\mathcal{Q}(t) = \sum_{i=1}^{q} \mathcal{Q}_{ij}(t)$. Let us consider that the energy consumption for running one bit of a task i at remote computing device is expressed as E_j^{CPU} (in Joules). Thus, the processing energy consumption of a task i while executing on the remote computing device j is defined as follows.

$$E_{ij}^{pro} = P_{ij} \times E_j^{CPU} \qquad : \forall i \in \mathcal{T}, j \in (\mathcal{F}, \mathcal{S}) \tag{8}$$

Similarly, the result of the task i is downloaded to the requested end device via local gateway. Consider hp_j as the channel power gain from remote computing device j while downloading the ith task to the end device. The achievable downloading rate (R_{ji}^{down}) at the end device can be expressed as $R_{ji}^{\text{down}} = B_{ji}^{out} \log_2\left(1 + \frac{\mathbb{P}_i^{\text{down}} hp_j}{\xi_j^2}\right)$, where B_{ji}^{out} signifies the obtainable bandwidth utilization between jth remote computing device to ith end device. $\mathbb{P}_i^{\text{down}}$ denotes the required transmission power to download the ith task; and ξ_j^2 is an additive white Gaussian noise on the jth computing device. The downloading time T_{ji}^{down} to the end device can be expressed as follows.

$$T_{ji}^{\text{down}} = \frac{\mathcal{X}(j,i) \times T_i^{out}}{R_{ji}^{\text{down}}} \quad : \forall i \in \mathcal{T}, j \in (\mathcal{F}, \mathcal{S}) \tag{9}$$

Consequently, the corresponding downloading energy consumption of task i is defined as follows.

$$\begin{aligned} E_{ji}^{\text{down}} &= T_{ji}^{\text{down}} \times \mathbb{P}_i^{\text{down}} : \forall i \in \mathcal{T}, j \in (\mathcal{F}, \mathcal{S}) \\ &= \frac{\mathcal{X}(j,i) \times T_i^{out} \times \mathbb{P}_i^{down}}{B_{ji}^{out} \times log_2\left(1 + \frac{\mathbb{P}_i^{down} \times hp_j}{\xi_j^2}\right)} \end{aligned} \tag{10}$$

Thus, the total energy consumed by the task i while processing in a remote computing device j is defined as follows.

$$E_{ij}^{\text{total}} = E_{ij}^{\text{up}} + E_{ij}^{pro} + E_{ji}^{\text{down}} \tag{11}$$

As a result, the total energy consumption by a task $i, \forall i \in \mathcal{T}$ in a computing device $j, \forall j \in (\mathcal{F} \cup \mathcal{S} \cup \mathcal{I})$ at time t is formulated as follows.

$$E_{ij}^{\text{total}}(t) = \begin{cases} E_{ij}^{pro}(t), & \text{if } j \in \mathcal{I} \\ E_{ij}^{\text{up}}(t) + E_{ij}^{pro}(t) + E_{ji}^{\text{down}}(t), & \text{if } j \in \mathcal{FS} \end{cases} \tag{12}$$

3.2 Problem Formulation

In this section, we formulate the problem of jointly optimizing the scheduling strategy and offloading decision in a hierarchical fog networks. Specifically, we aim for optimizing the expected time-average energy consumption for the task i, *i.e.*, energy consumption during uploading (E_{ij}^{up}), processing (E_{ij}^{pro}) and downloading (E_{ij}^{down}) time on the jth computing device, $j \in (\mathcal{F} \cup \mathcal{S} \cup \mathcal{I})$. The above-mentioned objective function along with the instigate constraints can be formulated as follows.

$$\text{minimize} \quad \lim_{t\to\infty} \sum_{t=1}^{T} E_{ij}^{\text{total}}(t) \tag{13a}$$

$$\text{subject to} \quad E_{ij}^{\text{total}}(t) \le \mathcal{E}_j^{max}, \quad j \in (\mathcal{F} \cup \mathcal{S}), \tag{13b}$$

$$\mathscr{Q}_{ij}(t) \le \mathscr{Q}_j^{max}, \quad j \in (\mathcal{F} \cup \mathcal{S}), \tag{13c}$$

$$f_i^{CPU}(t) \le f_j^{max}, \quad j \in (\mathcal{F} \cup \mathcal{S}), \tag{13d}$$

$$\sum_{i=1}^{|\mathcal{T}|} \sum_{j=1}^{|\mathcal{FS}|} \mathcal{X}(i,j) \le |\mathcal{FS}|, \tag{13e}$$

$$\sum_{i=1}^{|\mathcal{T}|} \mathcal{X}(i,j) = 1, \tag{13f}$$

$$\mathcal{X}(i,j) \in \{0,1\}, \tag{13g}$$

$$T_{ij}^{\text{up}} \ge 0 \text{ and } T_{ji}^{\text{down}} \ge 0 \tag{13h}$$

The optimization objective of the above problem is to mitigate the total energy consumption in a time frame t, which is addressed in (13a). The constraint (13b) states that the total energy consumption for processing ith task is less than or equal to its maximum value $\mathcal{E}_j^{max}$ on jth computing device, $\forall i \in \mathcal{T}$. The constraint (13c) is the limitation of total waiting time, where $\mathscr{Q}_j^{max}$ is the maximum waiting time of task i on jth computing device. The constraint (13d) clarify that the required CPU frequency for processing ith task is less than or equal to the maximum CPU frequency f_j^{max} of jth computing device $\forall j \in (\mathcal{I} \cup \mathcal{F} \cup \mathcal{S})$. Constraints (13e) prevents the overloading of available computing resources, where $|\mathcal{FS}|$ represents the active computing devices [27]. Constraint (13f) defines that each task must be assigned at-most one computing device at time t. Constraint (13g) imposes the binary offloading constraint, *i.e.*, $\mathcal{F} \cap \mathcal{S} = \phi$. Finally, the constraint (13h) signifies that the task uploading time and task downloading time should be greater than or equal to zero.

4 Energy Efficient Task Offloading (EETO)

In this section, we discuss the scheduling and offloading mechanisms of the proposed Energy-Efficient Task Offloading (EETO) policy with Lyapunov optimization technique. Initially, a heuristic-based queue assignment (QA) policy is introduced for assigning a priority on each incoming task according to its importance and deadline. Further, a modified Lyapunov optimization model with a utility function (*drift-plus-penalty* framework) has been designed, namely Optimal Task Scheduling (OTS) policy, for jointly minimizing energy consumption of the real-time tasks and stabilizing the queues. Finally, a Constrained Restriction Offloading (CRO) mechanism has been developed for offloading the scheduled tasks on the suitable remote computing devices in the fog network. The mechanisms of the proposed EETO strategy with a suitable workflow model are discussed as follows.

4.1 Queueing Assignment (QA) Policy

The inherent intuition of designing QA policy is to prioritize the arrival tasks based on their importance and allocate them to one of the priority queues of a local gateway. Consider a task $T_i = \langle C_i, D_i \rangle, T_i \in \mathcal{T}$, is characterized by execution requirement C_i and inter-arrival time difference D_i, also called period of a task. For a task i, the *utilization factor* is defined as follows.

$$\mathscr{U}(T_i) \stackrel{def}{=} \frac{C_i}{D_i} \tag{14}$$

Thus, the *cumulative utilization* factor $\mathscr{U}_{sum}(\mathcal{T})$ of a task set $\mathcal{T}$ is defined as $\mathscr{U}_{sum}(\mathcal{T}) = \sum_{i=1}^{q} \mathscr{U}(T_i)$. Further the largest utilization of any task T_i is derived as follows.

$$\mathscr{U}_{max}(\mathcal{T}) \stackrel{def}{=} \max_{1 \le i \le q} \mathscr{U}(T_i) \tag{15}$$

We also consider that tasks are sorted according to non-decreasing utilization: $\mathscr{U}(T_i) \ge \mathscr{U}(T_{i+1}), \forall i, 1 \le i \le q$. For rest of the paper, we consider *1)* $\mathscr{U}_{max}(\mathcal{T}) \le 1$ and *2)* we do not permit job level parallelism. Based on the utilization

Algorithm 1: EETO : QA Policy

INPUT : q : Number of tasks, C_i : Execution requirement, D_i : Task period.
OUTPUT : Priority queue assignment $\left(\mathcal{Q}_i^D, \mathcal{Q}_j^C\right)$.
begin
  for *i: 1 to q* **do**
    Calculate *utilization factor* $\mathscr{U}(T_i) = \frac{C_i}{D_i}$
    Assign priority to each task T_i
    if *utilization factor* $\mathscr{U}(T_i) > \frac{1}{2}$ **then**
      Classify task T_i as *delay-sensitive* task T_i^D
    end
    if *utilization factor* $\mathscr{U}(T_i) \leq \frac{1}{2}$ **then**
      Classify task T_i as *computation-intensive* task T_i^C
    end
    Initialize $\mathcal{Q}_i^D \leftarrow 0$ and $\mathcal{Q}_j^C \leftarrow 0$
    Assign tasks to a priority queue $\mathcal{Q}_i^D$ and $\mathcal{Q}_j^C$
    if *task priority* $T_i^D \leftarrow Ti$ **then**
      Enqueue task T_i into *delay-sensitive* queue $\mathcal{Q}_i^D$
    end
    if *task priority* $T_i^C \leftarrow Ti$ **then**
      Enqueue T_i into *computation-intensive* queue $\mathcal{Q}_j^C$
    end
    Set $i = i + 1$
  end
end

factor $\mathscr{U}(T_i)$, a task i can be classified as *delay-sensitive* or *computation-intensive*.

Definition 1. *A task T_i is called a delay-sensitive task if $\mathscr{U}(T_i) > \frac{1}{2}$, or a computation-intensive task if $\mathscr{U}(T_i) \leq \frac{1}{2}$.*

By doing so, we can highlight higher priority tasks, which essentially endeavor a faster reply in the fog networks. Note that, the priority of *delay-sensitive* and *computation-intensive* tasks are represented as T_i^D and T_i^C, respectively. In addition to that, the QA policy considers two priority queues as *delay-sensitive* queue ($\mathcal{Q}_i^D$) and *computation-intensive* queue ($\mathcal{Q}_j^C$) for keeping all the incoming tasks in a local gateway.

Definition 2. *A task T_i can be assigned to the delay-sensitive queue $\mathcal{Q}_i^D$, if $T_i \in T_i^D$ or to the computation-insensitive queue $\mathcal{Q}_i^C$, if $T_i \in T_i^D$.*

An illustration Example: Considering a task set $\mathcal{T}$ with five real-time tasks, $\mathcal{T} = \left(T_1, T_2, T_3, T_4, T_5\right)$, where $T_1 = \langle 2, 6\rangle$, $T_2 = \langle 5, 10\rangle$, $T_3 = \langle 3, 12\rangle$, $T_4 = \langle 4, 20\rangle$ and $T_5 = \langle 15, 20\rangle$. Now the utilization for T_1 is $\mathscr{U}(T_1) = \frac{2}{6} = 0.33$. Similarly for $\mathscr{U}(T_2) = 0.50$, $\mathscr{U}(T_3) = 0.25$, $\mathscr{U}(T_4) = 0.20$ and $\mathscr{U}(T_5) = 0.60$, respectively. For the set of tasks $\mathcal{T}$, if we consider the utilization threshold $\left(> \frac{1}{2}\right)$, then according to **Definition 1**, only task T_5 can be classifies as *delay-sensitive* task (as $0.60 > 0.50$) and assigned to *delay-sensitive* queue. Rest of the tasks are classified as *computation-intensive* tasks. This pseudo-code of priority evaluation and QA policy is shown in $Algorithm\ 1$. We analyse the run-time of *Algorithm* 1 using $big{-}O$ notation and other system model parameters, where q represents the number of tasks in the system.

4.2 Optimal Task Scheduling (OTS) Policy

In this work, we steer to obtain an optimal task scheduling policy for all priority-aware tasks $\mathcal{T}$ in a stipulated time period, while making the system strongly stable [30]. To achieve our desired output, we designed the Lyapunov optimization framework for online task scheduling and efficient offloading decision.

4.2.1 Lyapunov Optimization

The key principal of Lyapunov optimization technique is to create a set of virtual queues, which leverage the precision-based queue stability problem and ensures to minimize the average queue waiting time and energy consumption. To follow this intuition, we transform all inequality constraints into Lyapunov queue stability form. Let us consider that, $Q_i^D(t)$ and $Q_j^C(t)$ be the number of waiting tasks in *delay-sensitive* and *computation-intensive* queues at an instance t. At first, we divide all the tasks $\mathcal{T}$ with arrival rate $\lambda_i^D(t)$ and $\lambda_j^C(t)$ into two virtual queue $Q_i^D(t)$ and $Q_j^C(t)$, respectively based on $Algorithm\ 1$, $\forall i, i \in \{1, 2, \ldots, I\}$ and $\forall j, j \in \{1, 2, \ldots, J\}$, where each i, j follow the dynamics, which are represented as follows.

$$Q_i^D\left(t+1\right) = \max\left[Q_i^D(t) - \mu_i(t), 0\right] + \lambda_i^D(t) \tag{16}$$

$$Q_j^C\left(t+1\right) = \max\left[Q_j^C(t) - \mu_j(t), 0\right] + \lambda_j^C(t) \tag{17}$$

Here, queue backlog (*e.g.*, $Q_i^D(t)$ and $Q_j^C(t)$) depend on the difference between service rate and the total amount of requests satisfied for each time interval t.

4.2.2 Queue Stability

Let us consider that $L\left(\Theta\left(t\right)\right)$ be a Lyapunov function. To establish the virtual queue stability condition, we adopt Lyapunov quadratic-drift approach [31], which is defined as follows.

$$L\left(\Theta\left(t\right)\right) \triangleq \frac{1}{2}\sum_{i,j\in\mathcal{T}}\left[Q_i^D(t)^2 + Q_j^C(t)^2\right] \tag{18}$$

where, $L\left(\Theta\left(0\right)\right) = 0$ and $\Theta\left(t\right) \triangleq \left\{Q_i^D(t), Q_j^C(t)\right\}$, $\forall i, \forall j, (i,j) \in \mathcal{T}, i = \{1, 2, \ldots, I\}$ and $j = \{1, 2, \ldots, J\}$, represent the scalar number of queue congestion on a certain time instance t. Now, we can define both the state and stability of the queues at time instance t, which is derived as follows.

$$\lim_{t\to\infty} sup\ \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[Q_i^D(t) + Q_j^C(t)\right] < \infty \tag{19}$$

In a time slot t, a system is *strongly stable* if queues are individually stable with $\left(\mu_k\left(t\right) - \lambda_k(t)\right) \geq 0$, where $\mu_k\left(t\right), \forall k \in (i,j)$ represents the service rate, *i.e.*, $min\left[Q_i^D(t), \mu_i(t)\right]$ and $min\left[Q_j^C(t), \mu_j(t)\right]$ of the system. Further, we consider that initial queue backlog $Q_i^D(t) = 0$ and $Q_j^C(t) = 0$, when $t = 0$. From the Lyapunov function $L\left(\Theta\left(t\right)\right)$, we formulate the Lyapunov drift $\Delta\Theta\left(t\right)$ as the rate of change from $L\left(\Theta\left(t+1\right)\right)$ to $L\left(\Theta\left(t\right)\right)$ for next time frame, which is formulated as follows.

$$\Delta(\Theta\left(t\right)) \triangleq \mathbb{E}\Big[L\left(\Theta\left(t+1\right)\right) - L\left(\Theta\left(t\right)\right) \mid \Theta\left(t\right)\Big] \tag{20}$$

Theorem 1 : *Assuming that constraints $\mathcal{D} > 0$ and $\mathbb{E}\left\{L(\Theta\left(0\right))\right\} < \infty$. A quadratic Lyapunov-drift function satisfy a condition $\Delta\left(\Theta\left(t\right)\right) \leq \mathcal{D} - \alpha\sum_{n=1}^{N}\left|\Theta_n\left(t\right)\right|$ for all possible values of $\Theta\left(t\right)$, and for a positive constraint α, where $\alpha > 0$. Then, if a queuing system is stable, all the virtual queues are strongly stable and strictly follow $\alpha > 0$, $\forall t, t = \{1, 2, \ldots, T\}$.*

$$\lim_{t\to\infty} Sup\frac{1}{t}\sum_{t=0}^{T-1}\sum_{n=1}^{N}\mathbb{E}\Big\{\left|\Theta_n(t)\right|\Big\} \leq \frac{\mathcal{D}}{\alpha} \tag{21}$$

Algorithm 2: EETO:OTS Policy

INPUT : ϑ : Control parameter, $\mathcal{Q}_i^D$: *Delay-sensitive* queue, $\mathcal{Q}_i^C$: *Computation-intensive* queue, T_i^D : *Delay-sensitive* task, T_i^C : *Computation-intensive* task, q : Number of tasks.
OUTPUT : Optimal scheduling order $\Gamma(t)$.
begin
for *i: 1 to q* **do**
Assign tasks to virtual queues (Q_i^D, Q_i^D)
if *Task* $\mathcal{Q}_i^D \leftarrow T_i$ **then**
Assign task T_i to *delay-sensitive* virtual queue Q_i^D
end
if *Task* $\mathcal{Q}_i^D \leftarrow T_i$ **then**
Assign task T_i to *delay-sensitive* virtual queue Q_i^D
end
end
Initialize $Q_i^D(0) = 0$ and $Q_j^C(0) = 0$
Observe current queue backlog $Q_i^D(t)$ and $Q_j^C(t)$ at the beginning of time t
for *i: 1 to q* **do**
Find an optimal policy $\Gamma(t)$, $\forall T_i \in (\mathcal{Q}_i^D(t) \cup \mathcal{Q}_j^C(t))$
Get current system information **Determine** $\mathscr{Y}_{T_i}^{opt}(t)$ by minimizing $\vartheta \sum_{j\in\mathcal{F},\mathcal{S}}((T_i^{in}\mathbb{P}_j^{up})/B_{ij}^{in} + (E_j^{CPU}T_i^{in})/f_j^{CPU} + (T_i^{out}\mathbb{P}_i^{down})/B_{ji}^{out}) - \sum_{i\in I} Q_i^D(t)\mu_i(t) - \sum_{j\in J} Q_j^C(t)\mu_j(t)$
Subject to (13b)-(13h), (19)
Update virtual queue $Q_i^D(t)$ and $Q_j^C(t)$ according to (16) and (17);
Set $i = i + 1$;
end
$\Gamma(t) \leftarrow$ Values of $\mathscr{Y}_{T_i}^{opt}(t)$ at time t
end

Proof : Complete *proof* of this Theorem can be found in *Appendix* A.

Drift-Plus-Penalty: Here, we jointly satisfy the queue stability constrains and minimize the average energy consumption for all incoming tasks. As in [31], we define the *drift-plus-penalty* as $\Delta(\Theta(t)) + \vartheta \sum_{i\in\mathcal{T}} \mathbb{E}\left[E_{ij}^{total}(t) \mid \Theta(t)\right]$, where ϑ is a positive scalar limiting parameter used to control the trade-off between the backlog queues and energy consumption which minimizes the upper bound of the penalty [32]. Hence, the objective function of our work under the condition of queue stability constraints is transformed as follows.

$$\text{minimize} \quad \Delta(\Theta(t)) + \vartheta \sum_{i\in\mathcal{T}} \mathbb{E}\left[E_{ij}^{total}(t) \mid \Theta(t)\right] \tag{22a}$$

$$\text{subject to} \quad (13b) - (13h), (19) \tag{22b}$$

Conforming the Lyapunov optimization technique, the constraints diminishes the drift penalty $\Delta\Theta(t)$, and optimize the objective function with the corresponding constraints, which is identical to enhance the "*drift-plus-penalty*". From the Eq. (18) and Eq. (20), it is observed that, the objective function (22a) contains some variables for next time instance $t+1$, which are eradicated to find an optimal solution of the objective function. To follow this, we need to consider upper bound of $\left(Q_i^D(t+1)^2 - Q_i^D(t)^2\right)$ of the virtual queue.

Theorem 2 : *Assuming that $\lambda_i(t)$ follow independent and identically distributed (i.i.d) over time instant t. The upper bound of the Lyapunov-drift function over a random scheduling policy $\Gamma(t)$ and all possible values of $\Theta(t)$, $\forall t, t \in \{1, 2, \ldots, T\}$, is represented as follows.*

$$\begin{aligned}\mathbb{E}\Big[L(\Theta(t+1)) - L(\Theta(t))\Big] \leq \mathcal{D} + \sum_{i\in I} Q_i^D(t)\Big[\lambda_i^D(t) \\ -\mu_i(t)|\Theta(t)\Big] + \sum_{j\in J} Q_j^C(t)\Big[\lambda_j^C(t) - \mu_j(t)|\Theta(t)\Big]\end{aligned} \tag{23}$$

Where, $\mathcal{D}$ is a positive constant value and can be defined as follows.

$$\begin{aligned}\mathcal{D} = \frac{\lambda_i^D(t)^2 - \mu_i(t)^2}{2} + \frac{\lambda_j^C(t)^2 - \mu_j(t)^2}{2} \\ - \Big\{\lambda_i^D(t)\mu_i(t) + \lambda_j^C(t)\mu_j(t)\Big\}\end{aligned} \tag{24}$$

Proof : Complete *proof* of this Theorem can be found in *Appendix* B.

Now adding penalty function $\vartheta \sum_{i\in\mathcal{T}} \mathbb{E}\Big[E_{ij}^{total}(t)|\Theta(t)\Big]$ on both the side of (23) yields the upper bound of *drift-plus-penalty*, which is formulated as follows.

$$\begin{aligned}\mathbb{E}\Big[L(\Theta(t+1)) - L(\Theta(t))\Big] + \vartheta \sum_{i\in\mathcal{T}} \mathbb{E}\Big[E_{ij}^{total}(t)|\Theta(t)\Big] \\ \leq \mathcal{D} + \vartheta \sum_{i\in\mathcal{T}} \mathbb{E}\Big[E_{ij}^{total}(t)|\Theta(t)\Big] + \sum_{i\in I} Q_i^D(t)\Big[\lambda_i^D(t) \\ -\mu_i(t)|\Theta(t)\Big] + \sum_{j\in J} Q_j^C(t)\Big[\lambda_j^C(t) - \mu_j(t)|\Theta(t)\Big]\end{aligned} \tag{25}$$

Form the above formulations, we observe that the right hand side of Eq. (25) depends only on the variables at time t, *i.e.* the upper-bound is restricted on t. Hence, the optimization problem becomes the *drift-plus-optimization* with associated upper bound. The motivated constraints are designed in *drift-plus-penalty* with respect to the queue stability function.

The development concept of our OTS policy is now transformed into minimizing the upper bound of *drift-plus-penalty* contrary to meet the constraints (13d) to (13h) as discussed in subsection 3.2. Given $\Theta(t)$ and set of constraints in each time instance t, we can find an upper bound of the *drift-plus-penalty* by minimizing the right hand side of Eq. (25), which is derived as follows.

$$\text{minimize}\Big\{\mathcal{D} + \mathbb{E}(\phi) + \vartheta \sum_{j\in(\mathcal{F},\mathcal{S})} \mathbb{E}\left[E_{ij}^{total}(t) \mid \Theta(t)\right]\Big\} \tag{26}$$

where, $E_{ij}^{total} = T_i^{in}\mathbb{P}_j^{up}/\Big(B_{ij}^{in} log_2(1 + \frac{P_j^{up}\times hp_i}{\xi_i^2})\Big) + P_{ij}E_j^{CPU} + T_i^{out}\mathbb{P}_i^{down}/\Big(B_{ji}^{out} log_2(1 + \frac{\mathbb{P}_i^{down}\times hp_j}{\xi_j^2})\Big) + P_{ij}E_j^{CPU}$ and $\phi = \sum_{i\in I} Q_i^D(t)\Big[\lambda_i^D(t) - \mu_i(t)|\Theta(t)\Big] + \sum_{j\in J} Q_j^C(t)\Big[\lambda_j^C(t) - \mu_j(t)|\Theta(t)\Big]$.

From (26), we observe that the upper bound of *drift-plus-penalty* depends on both variables and constants. We acquire minimum values by considering the variables and skipping the constants from the penalty function. Hence, the updated minimized objective function is represented as follows.

$$\begin{aligned}\text{minimize}\ \Big\{\mathbb{E}\Big(\sum_{i\in I} Q_i^D(t)\Big[\lambda_i^D(t) - \mu_i(t)|\Theta(t)\Big] + \\ \sum_{j\in J} Q_j^C(t)\Big[\lambda_j^C(t) - \mu_j(t)|\Theta(t)\Big] + \vartheta \sum_{j\in\mathcal{F},\mathcal{S}} \Big((T_i^{in}\mathbb{P}_j^{up})/ \\ B_{ij}^{in} + (E_j^{CPU}T_i^{in})/f_j^{CPU} + (T_i^{out}\mathbb{P}_i^{down})/B_{ji}^{out}\Big)\Big)\Big\}\end{aligned} \tag{27}$$

Algorithm 3: EETO : CRO Policy

INPUT : q : Set of tasks, k : Set of computing devices, $\Gamma(t)$: Scheduling order.
OUTPUT : Suitable matching order $\mathscr{K}(t)$.
begin
  $\mathcal{T} \leftarrow \{T_i^D \cup T_i^C\}, \mathcal{FS} \leftarrow \{\mathcal{F} \cup \mathcal{S}\}$
  if $|\mathcal{FS}| = \phi$ **then**
    Wait for $\mathcal{FS}$
  end
  for i= *1 to* q **do**
    for j= *1 to* k **do**
      if $i \leq |\mathcal{T}|, j \leq |\mathcal{FS}|$ *and* $T_i \in T_i^D$ **then**
        Assign task T_i to $\mathcal{F}_j$ by satisfying constraints (13b), (13c) and (13d)
      end
      if $i \leq |\mathcal{T}|, j \leq |\mathcal{FS}|$ *and* $T_i \in T_i^C$ **then**
        Assign task T_i to $\mathcal{S}_j$ by satisfying constraints (13b), (13c) and (13d)
      end
      $\mathcal{T} \leftarrow \mathcal{T} \backslash \{i\}$ and $\mathcal{FS} \leftarrow \mathcal{FS} \backslash \{j\}$
      $\mathscr{K}(t) \leftarrow$ Task assignment value at time t
    end
  end
end

In order to find a near optimal solution, a large positive ϑ, and $E_{ij}^{pro} > 0, \forall j \in (\mathcal{F} \cup \mathcal{S})$ values are preferable. OTS policy includes both updating virtual queues and optimal scheduling order in each time stamp t to produce a stable and efficient output.

Task Offloading Decision : If we look closely, then we can analyze that the term $\sum_{j \in J} Q_i^D(t)\lambda_i^D(t)$ and $\sum_{j \in J} Q_j^C(t)\lambda_j^C(t)$ do not have any impact on offloading and downloading decisions at time t. Thus, we drive the energy-efficient task offloading decisions by determining the following function.

$$\begin{aligned}\mathscr{Y}_{T_i}^{opt}(t) = \text{minimize} \Big\{ \vartheta \sum_{j \in \mathcal{F},\mathcal{S}} \Big((T_i^{in}\mathbb{P}_j^{up})/B_{ij}^{in} \\ +(E_j^{CPU}T_i^{in})/f_j^{CPU} + (T_i^{out}\mathbb{P}_i^{down})/B_{ji}^{out} \Big) \\ - \sum_{i \in I} Q_i^D(t)\mu_i(t) - \sum_{j \in J} Q_j^C(t)\mu_j(t) \Big\} \\ \text{Subject to} \quad (13b) - (13h), (19)\end{aligned} \tag{28}$$

According to constraint (13h), optimized Eq. (28) returns a positive value, which can be obtained by tuning the best set of hyper-parameters. Moreover, Eq. (28) contains two results for remote offloading, *i.e.* either offload the tasks to the local fog nodes or centralized cloud servers. The pseudo-code of the proposed OTS policy is derived in *Algorithm* 2.

4.3 Constraint Restricted Offloading (CRO) Policy:

The main purpose of the CRO policy is to offload the q number of scheduled tasks (where $q = |\mathcal{T}|$) to the k number of remote computing devices (where $k = (|\mathcal{F}| + |\mathcal{S}|)$) as per the offloading decision obtained by *Algorithm* 2 using Eq. (28). The CRO policy primarily contains two *stages* for satisfying QoS constraints of the proposed fog networks such as constraints (13b), (13c) and (13d). $Stage.1$ helps to offload the *delay-sensitive* tasks to the local fog devices and $Stage.2$ uses for offloading the *computation-intensive* tasks to the centralized cloud servers. This can minimize the overall waiting time and offloading delay for the incoming tasks while meeting the energy and delay constraints of the fog networks. The pseudo-code of the proposed CRO policy is depicted in *Algorithm* 3.

In $step - 1$, the proposed CRO policy initializes both the scheduled tasks and active computing devices. The algorithm starts with the condition $|\mathcal{T}| \leq |\mathcal{FS}|$, *i.e.* $q \leq k$. Then, the CRO policy inducts the process and starts offloading the tasks based on their priority and the offloading decision obtained using Eq. (28). $step - 6$ to $step - 7$ of *Algorithm* 3 are used to offload the *delay-sensitive* tasks on the local fog devices, whereas the $step - 8$ to $step - 9$ are used for offloading the *computation-intensive* tasks on the centralized cloud servers while meeting the QoS constraints. In $step - 9$, unsuccessful offloading tasks are supervised by forwarding to the next timestamp $t + 1$. This process continues until all the tasks are offloaded to the suitable computing devices. In the next Section 4.4, we demonstrate that if Eq. (28) is optimized in the subsequent time frame, then a quantified near-optimal solution can be achieved.

Theorem 3: *The worst case run-time complexity of* EETO *strategy is* $O(qk)$, *where* $q = |\mathcal{T}|$ *and* $k = (|\mathcal{F}| + |\mathcal{S}|)$.

Proof: The proposed EETO strategy provides an online workload distribution strategy for the set of scheduled tasks in fog networks. At first, the EETO strategy initiates QA policy (*i.e. Algorithm* 1) to prioritize the arrival tasks. The process of task priority assignment policy depends on the "**for**" loop in *Algorithm* 1 and starts with by calculating the *utilization factor* of q number of tasks (in $step - 2$). Once the utilization factor is known for each task, the priority assignment can be determined (from $step - 3$ to $step - 7$) in linear time, *i.e.,* $O(1)$. Next, each task is *enqueue* to a suitable queue according to its priority (refer to $step - 8$ to $step - 12$), which takes $O(1)$ time. Thus, the total run-time complexity of each task is $O(1) + O(1) = 2 \times O(1) = O(1)$. The total run-time complexity of QA policy with q number of tasks is $q \times O(1) = O(q)$. Similar to *Algorithm* 1, the EETO strategy renders an OTS algorithm (*i.e. Algorithm* 2) for updating the virtual queues and schedules the tasks in each time frame with an offloading decision. $step - 1$ to $step - 6$ of *Algorithm* 2 updates the virtual queues of the gateway with q number of priority-aware tasks, which takes $O(q)$ run-time complexity. Next, $step - 9$ to $step - 15$ determines the scheduling order and takes task offloading decision, which takes $O(q)$ run-time complexity. Thus the total run-time complexity of OTS policy is $O(q) + O(q) = 2O(q) = O(q)$. Finally, the CRO policy (*i.e. Algorithm* 3) helps to offload q number of scheduled tasks on the k number of suitable remote computing devices, where $k = (|\mathcal{F}| + |\mathcal{S}|)$. Thus, the run-time complexity of the CRO policy is $O(qk)$. The total run-time complexity of EETO strategy is $O(q) + O(q) + O(qk) = 2O(q) + O(qk)$. As $qk >> q$, thus the total run-time complexity of EETO strategy is $O(qk)$. The workflow of the proposed EETO strategy is depicted in Fig. 2.

4.4 Theoretical Analysis

The proposed EETO strategy is used for finding a feasible scheduling order with an efficient offloading decision in

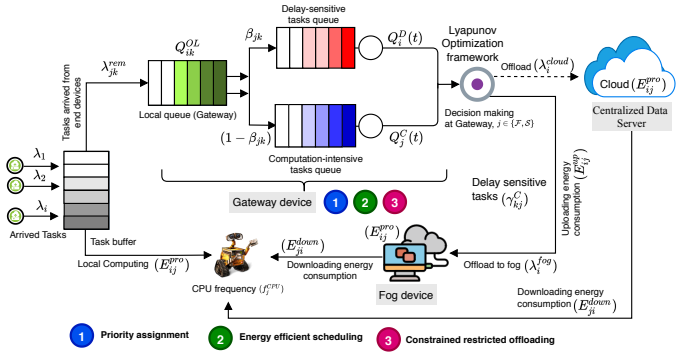

Fig. 2. Workflow of EETO strategy

fog networks, which requires current system information in each time frame without prior system dynamics. This nature of the proposed algorithm makes it much easier to implement for a fog network with low complexity. Using the Lyapunov optimization technique, we determine the upper bound of the proposed EETO strategy.

Theorem 4 : *Assuming λ_i be the task arrival rate, if a system is stable under λ_i and μ_i with optimal offloading decision, then the upper bound of the EETO strategy in terms of energy consumption and average queue waiting time are derived as follows.*

$$\lim_{t\to\infty} Sup \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\left[E_{ij}^{\text{total}}(t)\right] \leq E^{opt} + \frac{D}{\vartheta} \tag{29}$$

$$\lim_{t\to\infty} Sup \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\left[Q_i^D(t) + Q_i^C(t)\right] \leq \frac{1}{\varepsilon}\left(D + \vartheta \mathcal{P}\right) \tag{30}$$

where ϑ and $\mathcal{P}$ denote the penalty function and long term energy consumption rate achieved by the system.

Proof : Complete *proof* of this Theorem can be found in *Appendix* C.

Theorem 4 demonstrates the conflicting trade-off $\left[O\left(\vartheta\right), O\left(1/\vartheta\right)\right]$ between the upper bound of energy consumption $O\left(1/\vartheta\right)$ and average queue waiting time $O\left(\vartheta\right)$. Theorem 4 indicates the response discrepancy between Eq. (13a) and *Algorithm* 2, which can diminish by adjusting the control parameter ϑ. If ϑ is increasing, it will generate more substantial backlog of queues with more energy usage. Thus, the approximate value of ϑ is needed to satisfy the requirements of the test in the practical implementation. It can be noted that the Theorem 2 and Theorem 4 represent the feasibility of the proposed model with a stable energy consumption and queue waiting time.

5 Performance Evaluation

In this section, we briefly quantify the performance of the proposed EETO strategy in regard to a) *average queue waiting time*, b) *average task offloading delay*, c) *Average energy consumption* and d) *Throughput*. Here, we adopt a new SIMUL8 PROFESSIONAL simulator and consider a hierarchical fog network. Further, we compare our proposed EETO strategy with the two baseline algorithms [28] concerning various performance matrices, which are discussed briefly as follows.

TABLE 3
Parameters Used For Simulation

Parameters	Values
Total number of end devices ($\mathcal{I}$)	100
Total number of fog devices ($\mathcal{F}$)	40
Total number of cloud servers ($\mathcal{S}$)	10
Total number of gateway devices ($\mathcal{G}$)	5
Average number of incoming tasks (λ_i)	100 [tasks/sec]
Maximum channel bandwidth (B_{ij}^{in})	30 MHz
Task offloading probability ($\alpha_{jk}, \beta_{jk}, \gamma_{kj}$)	0.5
CPU frequency in end devices (f_j^{CPU})	500×10^6 [cyc/sec]
CPU frequency in fog devices (f_j^{CPU})	50×10^9 [cyc/sec]
CPU frequency in cloud servers (f_j^{CPU})	100×10^9 [cyc/sec]
Arrival rate for all tasks (λ_i)	[.1, .2, . . . , .9]
Control parameter (ϑ)	[500,. . . , 3000]
Processing energy usage (E_j^{CPU})	0.5 Joules
Transmission power of end devices ($\mathbb{P}_j^{\text{up}}$)	1 mW

- *Random Task Offloading (RTO)*: In the RTO strategy, the real-time tasks are handled locally or offloaded to a *randomly* selected computing devices without concerning the priority of the tasks.
- *Higher Transmission-Rate Offloading (HTRO)*: In the HTRO strategy, the real-time tasks are offloaded to the computing devices having higher data *transmission* capability instead concerning about the latency and energy usage.

To demonstrate the superiority of the proposed EETO strategy, we have compared the proposed strategy with there state-of-the-art algorithms including DPTO [28], evolutionary MOO [25], and Lyapunov-based CEAO [15] algorithms along with two baseline algorithms.

5.1 Experimental Simulation Setup

The entire simulation conducted on *Intel Core i7-2600 CPU @ 3.40GHz × 8* with *10GB RAM* using *Ubuntu 18.04.3 LTS* operating system. Table 3 contains a brief description of the simulation parameters [28]. Here, we consider 100 end devices that generate tasks randomly with the arrival rate of 100 tasks/sec [25]. For this experiment, we consider maximum data transmission rate 2.5 Mb/s, T_i^{in}=[50kb-10Mb], λ_i^{fog}=.125 and λ_i^{cloud}=.25. To make this simulation more realistic and achieve a near ground solution, we create additional assumptions into this model. In this network, out of 100 end devices, 20 end devices process the real-time tasks locally, and the remaining 80 devices offload their tasks to the distributed fog devices or centralize cloud servers for further processing through gateways. Further, we presume that the equal arrival frequency for task offloading from the end devices to the local fog devices or centralized cloud servers, *i.e.* $\gamma_{kj}^D = \gamma_{kj}^C = 0.5$. An experimental simulation is conducted for 1000 individual runs to make a feasible and stable solution with all varying parameters.

5.2 Average Queue Waiting Time

This parameter describes how long the tasks are waited before offloading and being allocated to a suitable computing device. Fig. 3(a) reflects the variation of the average queue waiting time of the proposed EETO strategy for *delay-sensitive* and *computation-intensive* tasks. Moreover, we consider that the average task arrival rate (λ_i) is 100 tasks/sec,

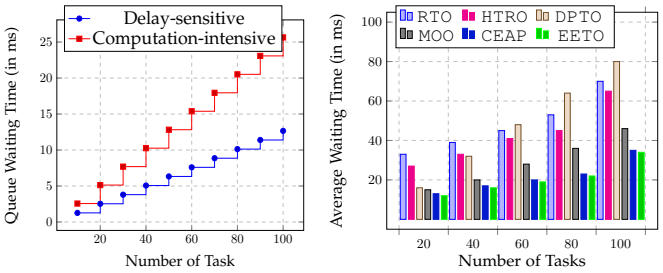

Fig. 3. Average queue waiting time (a)*delay-sensitive* and *computation-intensive* tasks; (b)EETO strategy and existing algorithms.

and all queues follow the FCFS order while keeping the infinite queue length. It is clear from Fig. 3(a) that normalized average queue waiting time for the *delay-sensitive* tasks is less as compared to the *computation-intensive* tasks due to their higher importance. Fig. 3(b) represents the comparison of the average queue waiting time of the proposed EETO strategy with the baseline and state-of-the art algorithms. The existing algorithms schedule the incoming tasks without considering their importance and assign the tasks on the suitable computing devices in a fog network without considering a suitable scheduling scheme, which causes higher queue waiting time and transmission delay. However, the proposed EETO strategy used the Lyapunov optimization technique for an efficient offloading and scheduling decision of the incoming real-time tasks, which minimizes the overall queue waiting time. The improvement of the proposed EETO strategy in term of average queue waiting time is 61.53%, 52.38%, 24.36%, 18.25%, and 16.66% over RTO, HTRO, DPTO, MOO and CEAP algorithm, respectively.

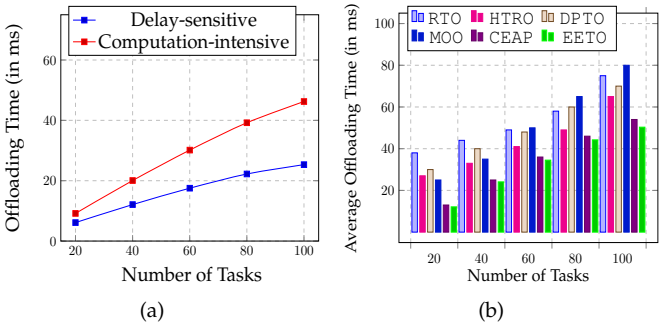

Fig. 4. Average task offloading time (a)*delay-sensitive* and *computation-intensive* tasks; (b)EETO strategy and existing algorithms.

5.3 Average Task Offloading Delay

This parameter represents the total task processing, uploading, and downlink time to/from the selected computing devices in fog network. The uploading and downloading time of each task depends on the available transmission bandwidth of the network. The offloading time of each task increases with decreasing the bandwidth availability in the network. Fig. 4(a) depicts the variation of normalized task offloading time between *delay-sensitive* and *computation-intensive* tasks in distributed fog networks. From the Fig. 4(a), it is observed that the task offloading time of the *delay-sensitive* tasks is lower than the *computation-intensive* tasks. The main reason behind that the proposed EETO strategy prefers to offload the *delay-sensitive* tasks through a gateway in the fog devices than the *computation-intensive* tasks due to their importance. Fig. 4(b) represents the performance comparison of the proposed EETO strategy over the baseline and state-of-the-art algorithms in terms of average offloading time. Fig. 4(b) depicts that the proposed EETO strategy minimizes the overall offloading time as compared to the existing algorithms. In general, the proposed EETO strategy saves more than 53.94%, 45.31%, 22.65%, 18.7% and 2.77% offloading time as compared with the RTO, HTRO, DPTO, MOO and CEAP algorithms, respectively.

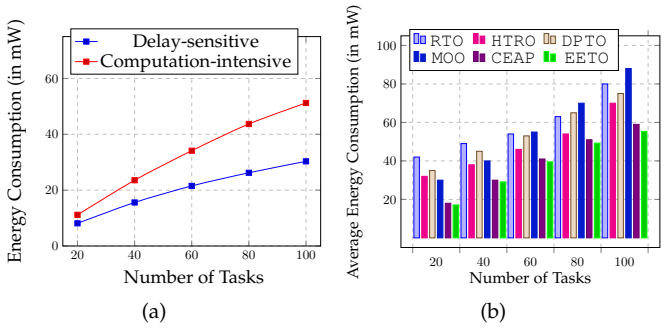

Fig. 5. Average energy consumption (a)*delay-sensitive* and *computation-intensive* tasks; (b)EETO strategy and existing algorithms.

5.4 Average Energy Consumption

This parameter represents the energy consumption rate for uploading, processing, and downloading of the tasks. Energy consumption largely relies on the CPU frequency of a computing device, and bandwidth of the network. Furthermore, this parameter also depends on the offloading rate of computing devices, which is shown in Fig. 5. Fig. 5(a) shows the comparison between the total energy consumption of various priorities of tasks (*i.e., delay-sensitive* and *computation-intensive*) with various numbers of input data and also depicts that the energy consumption rate increases while varying the workload in the network. Fig. 5(b) represents the comparison of energy consumption between the proposed EETO strategy and the baseline algorithms by varying the numbers of incoming tasks. Most of the baseline and state-of-the-art algorithms use various network parameters for finding a suitable computing device for the scheduled tasks, which do not help to minimize the energy consumption of the fog network. However, the proposed EETO strategy has achieved a new baseline mark for reducing the energy consumption by minimizing the total queue waiting time, as depicted in Fig. 3(b), and lower offloading delay, as shown in Fig. 4(b). This help to reduce the overall energy consumption, which is shown Fig. 5(b). The proposed EETO strategy saves more than 39.44%, 31.87%, 15.5%, 10.6% and 8.40% energy consumption as compared with the RTO, HTRO, DPTO, MOO and CEAP algorithm, respectively.

Fig. 6(a) represents the comparison of backlog queue processing among random scheduling random scheduling, Lyapunov drift, and without Lyapunov drift. From Fig. 6(a) it is noteworthy to say that by using Lyapunov drift function

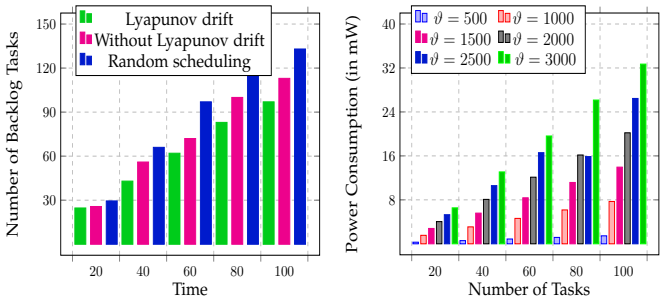

Fig. 6. Impact of Lyapunov drift (a)comparison with existing scheduling schemes; (b)comparison with different ϑ value.

into the scheduling strategy maintains the stability of the virtual queues, whereas random and without Lyapunov drift scheduling strategies imposes higher penalty, which mainly leads to increase in waiting time. Moreover, the total backlog queue increases by increasing the number of tasks and distance between end devices and remote computing devices. Fig. 6(b) represents the impact of energy consumption using the control parameter ϑ of *drift-plus-penalty* of the Lyapunov optimization technique. Fig. 6(b) also measures the relationship between the control parameter ϑ of the proposed EETO strategy with different time instance t. Furthermore, it shows that the energy consumption ($\times 10^3$) fluctuates in a short-range for small values of ϑ, but increases while increasing the value of ϑ. This illustrates that the control parameter ϑ directly impacts on the time instance of the tasks while satisfying the constraints.

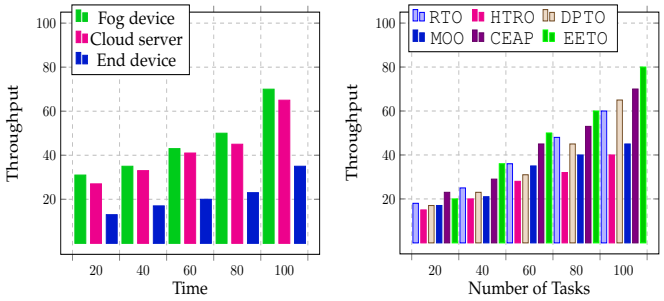

Fig. 7. Performance analysis in (a)various computing devices; (b)EETO strategy and existing algorithms.

5.5 Throughput

This parameter signifies the total number of tasks that completed their processing in a stipulated time period. Fig. 7(a) reflects the variation of the number of completed tasks in different computing devices. From Fig. 7(a), it is clear that the maximum number of tasks have completed their processing in local fog devices in each time instance t than the centralized cloud servers. Similarly, we have presented the comparison of performance analysis between EETO strategy and state-of-the-art algorithms in Fig. 7(b). Further, we have tested this simulation 1000 times with various set of of real-time tasks for producing a stable result. Out of 100 tasks, fog devices complete 78 tasks while cloud servers complete 22 number of tasks, *i.e.* 78% of tasks are processed in local fog devices and 28% of tasks are assigned on resource-rich cloud servers. Furthermore, among the total available resources, the fog devices utilize 85-95% of computing resources. However, centralized cloud servers utilize 20-30% of the resources while processing the real-time tasks.

6 Conclusion

This paper studies an energy-aware optimization framework, called EETO strategy for minimizing queue waiting time and energy consumption rate of the real-time tasks in fog networks. The proposed EETO strategy jointly classifies the tasks according to multiple QoS constraints and takes an efficient offloading decision using the Lyapunov optimization framework. The contributions of the work are three folded. Firstly, a queueing-based policy is designed for classifying the incoming tasks and assign them to the multiple priority queues in the local gateway of the fog network. Secondly, the Lyapunov optimization technique is used to obtain an efficient scheduling policy for the priority-aware tasks with the *drift-plus-penalty* function. Finally, we have designed a constrained restricted offloading policy for efficient offloading of the scheduled tasks on the suitable computing devices with low computational complexity. Thus, the proposed EETO strategy minimizes the average queue waiting time and total energy consumption of the real-time tasks on fog networks with higher throughput. The experimental analysis demonstrates that the proposed EETO strategy outperforms the baseline and existing algorithms in terms of average queue waiting time and energy consumption rate by 44.48% and 23.79%, respectively.

In future work, we will extend our work in the following directions: (1) optimizing multiple QoS parameters for taking an efficient offloading decision in the fog networks without considering any global information; and (2) design an optimal scheduling policy under different resource configurations with a combination of reward-penalty.

References

[1] K. Wang, Y. Shao, L. Xie, J. Wu, and S. Guo, "Adaptive and fault-tolerant data processing in healthcare IoT based on fog computing," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 1, pp. 263–273, 2020.

[2] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, "A Survey on Internet of Things: Architecture, Enabling Technologies, Security and Privacy, and Applications," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1125–1142, 2017.

[3] H. Hao, C. Xu, M. Wang, L. Zhong, and D. O. Wu, "Stochastic Cooperative Multicast Scheduling for Cache-Enabled and Green 5G Networks," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE, 2019, pp. 1–6.

[4] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog Computing and Its Role in the Internet of Things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, 2012, pp. 13–16.

[5] S. Ghosh, A. Mukherjee, S. K. Ghosh, and R. Buyya, "Mobi-IoST: Mobility-aware Cloud-Fog-Edge-IoT Collaborative Framework for Time-Critical Applications," *IEEE Transactions on Network Science and Engineering*, 2019.

[6] L. Pu, X. Chen, J. Xu, and X. Fu, "D2D Fogging: An Energy-Efficient and Incentive-aware Task Offloading Framework via Network-assisted D2D Collaboration," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3887–3901, 2016.

[7] J. Xu, L. Chen, and P. Zhou, "Joint Service Caching and Task Offloading for Mobile Edge Computing in Dense Networks," in *IEEE INFOCOM Conf. on Computer Comm.* IEEE, 2018, pp. 207–215.

[8] J. Zhang, X. Hu, Z. Ning, E. C.-H. Ngai, L. Zhou, J. Wei, J. Cheng, and B. Hu, "Energy-Latency Tradeoff for Energy-aware Offloading in Mobile Edge Computing Networks," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2633–2645, 2017.

[9] L. Chen, S. Zhou, and J. Xu, "Computation Peer Offloading for Energy-Constrained Mobile Edge Computing in Small-Cell Networks," *IEEE/ACM Transactions on Networking*, vol. 26, no. 4, pp. 1619–1632, 2018.

[10] L. Gu, J. Cai, D. Zeng, Y. Zhang, H. Jin, and W. Dai, "Energy Efficient Task Allocation and Energy Scheduling in Green Energy Powered Edge Computing," *Future Gen. Com. Sys.*, vol. 95, pp. 89–99, 2019.

[11] Y. Yang, K. Wang, G. Zhang, X. Chen, X. Luo, and M.-T. Zhou, "MEETS: Maximal Energy Efficient Task Scheduling in Homogeneous Fog Networks," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 4076–4087, 2018.

[12] Y. Liu, F. R. Yu, X. Li, H. Ji, and V. C. Leung, "Hybrid Computation Offloading in Fog and Cloud Networks with Non-orthogonal Multiple Access," in *IEEE INFOCOM 2018-IEEE Conf. on Computer Comm. Workshops*. IEEE, 2018, pp. 154–159.

[13] G. Wang, W. Cai, Y. Zhang, K. Zhao, and X. Xu, "Lyapunov Optimization Based Online Energy Flow Control for Multi-energy Community Microgrids," in *2019 IEEE PES GTD Grand International Conference and Exposition Asia (GTD Asia)*. IEEE, 2019, pp. 706–711.

[14] T. Mori, Y. Utsunomiya, X. Tian, and T. Okuda, "Queueing theoretic approach to job assignment strategy considering various inter-arrival of job in fog computing," in *2017 19th Asia-Pacific Network Operations and Management Symposium (APNOMS)*. IEEE, 2017, pp. 151–156.

[15] S. Pan and Y. Chen, "Energy-Optimal Scheduling of Mobile Cloud Computing Based on a Modified Lyapunov Optimization Method," *IEEE Transactions on Green Communications and Networking*, vol. 3, no. 1, pp. 227–235, 2019.

[16] X. Wang, K. Wang, S. Wu, S. Di, H. Jin, K. Yang, and S. Ou, "Dynamic Resource Scheduling in Mobile Edge Cloud with Cloud Radio Access Network," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 11, pp. 2429–2445, 2018.

[17] M. Mukherjee, M. Guo, J. Lloret, R. Iqbal, and Q. Zhang, "Deadline-Aware Fair Scheduling for Offloaded Tasks in Fog Computing With Inter-Fog Dependency," *IEEE Communications Letters*, vol. 24, no. 2, pp. 307–311, 2019.

[18] L. Li, Q. Guan, L. Jin, and M. Guo, "Resource Allocation and Task Offloading for Heterogeneous Real-time Tasks with Uncertain duration time in a Fog Queueing System," *IEEE Access*, vol. 7, pp. 9912–9925, 2019.

[19] Q. Fan and N. Ansari, "Workload Allocation in Hierarchical cloudlet Networks," *IEEE Comm. Letters*, vol. 22, no. 4, pp. 820–823, 2018.

[20] J. Kumar, A. Malik, S. K. Dhurandher, and P. Nicopolitidis, "Demand-based Computation Offloading Framework for Mobile Devices," *IEEE Systems Journal*, vol. 12, no. 4, pp. 3693–3702, 2017.

[21] L. Liu, Z. Chang, X. Guo, S. Mao, and T. Ristaniemi, "Multiobjective Optimization for Computation Offloading in Fog Computing," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 283–294, 2017.

[22] Y. Nan, W. Li, W. Bao, F. C. Delicato, P. F. Pires, Y. Dou, and A. Y. Zomaya, "Adaptive Energy-aware Computation Offloading for Cloud of Things Systems," *IEEE Access*, vol. 5, pp. 23 947–23 957, 2017.

[23] S. Pan and Y. Chen, "Energy-Optimal Scheduling of Mobile Cloud Computing Based on a Modified Lyapunov Optimization Method," *IEEE Transactions on Green Communications and Networking*, vol. 3, no. 1, pp. 227–235, 2018.

[24] M. Adhikari and H. Gianey, "Energy efficient offloading strategy in fog-cloud environment for IoT applications," *Internet of Things*, vol. 6, p. 100053, 2019.

[25] M. Adhikari, S. N. Srirama, and T. Amgoth, "Application Offloading strategy for Hierarchical Fog Environment Through Swarm Optimization," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4317–4328, 2019.

[26] C. Qiu, Y. Hu, Y. Chen, and B. Zeng, "Lyapunov Optimization for Energy Harvesting Wireless Sensor Communications," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1947–1956, 2018.

[27] M. Liwang, Z. Gao, S. Hosseinalipour, and H. Dai, "Multi-task offloading over vehicular clouds under graph-based representation," *arXiv preprint arXiv:1912.06243*, 2019.

[28] M. Adhikari, M. Mukherjee, and S. N. Srirama, "DPTO: A Deadline and Priority-aware Task Offloading in Fog Computing Framework Leveraging Multi-level Feedback Queueing," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 5773–5782, 2019.

[29] X. Wei, C. Tang, J. Fan, and S. Subramaniam, "Joint Optimization of Energy Consumption and Delay in Cloud-to-Thing Continuum," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2325–2337, 2019.

[30] G. Zhang, W. Zhang, Y. Cao, D. Li, and L. Wang, "Energy-Delay Tradeoff for Dynamic Offloading in Mobile-Edge Computing System With Energy Harvesting Devices," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4642–4655, 2018.

[31] M. J. Neely, "Stochastic Network Optimization with Application to Communication and Queueing systems," *Synthesis Lectures on Communication Networks*, vol. 3, no. 1, pp. 1–211, 2010.

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

Abhishek Hazra currently pursuing Ph.D. in IIT(ISM) Dhanbad, India. He completed his master's degree in Computer Science and Engineering from NIT Manipur, India in 2018 and Bachelor degree from NIT Agartala, India in 2014. He has authored and co-authored various national and international journal and conference articles. His research area of interest is in the field of Fog computing and Industrial Internet of Things.

Mainak Adhikari is currently working as a Post Doctorate Research Fellow at University of Tartu, Estonia. He has completed his Ph.D in Cloud Computing from IIT(ISM) Dhanbad, India in 2019. He has obtained his M.Tech. from Kalyani University in the year 2013. He earned his B.E. Degree from West Bengal University of Technology in the year of 2011. He is the Associate Editor of Cluster Computing Journal and IEEE IoT Magazine and Technical Committee member of Computer Communication Journal.

Tarachand Amgoth received B.Tech in Computer Science and Engineering from JNTU, Hyderabad and M.Tech in Computer Science Engineering from NIT, Rourkela in 2002 and 2006 respectively and Ph.D. form IIT(ISM), Dhanbad in 2015. Presently, he is working as an Assistant professor in the Department of Computer Science and Engineering, IIT(ISM), Dhanbad. His current research interest includes Fog/Edge computing, and Internet of Things.

Satish Narayana Srirama is an Associate Professor at School of Computer and Information Sciences, University of Hyderabad, India. He was a Research Professor and the head of the Mobile & Cloud Lab at the Institute of Computer Science, University of Tartu, Estonia. His current research focuses on cloud computing, mobile cloud, IoT, Fog computing, migrating scientific computing and large scale data analytics to the cloud. He received his PhD in computer science from RWTH Aachen University in 2008. He is an IEEE Senior Member and an Editor of Wiley Software: Practice and Experience journal.