DPTO: A Deadline and Priority-aware Task Offloading in Fog Computing Framework Leveraging Multi-level Feedback Queueing

Mainak Adhikari, Mithun Mukherjee, *Member, IEEE*, and Satish Narayana Srirama, *Senior Member, IEEE*

***Abstract*—By providing flexible and shared computing and communication resources along with cloud services, fog computing became an attractive paradigm to support delay-sensitive tasks in Internet of Things (IoT). Existing researches for offloading delay-sensitive tasks in a hierarchical fog-cloud environment mostly focused on minimizing the overall communication delay. However, a fair offloading strategy selects a suitable computing device in terms of fog node or cloud server based on the resource requirements of the task while meeting the deadline. In this paper, we design a new delay-dependent priority-aware offloading (DPTO) strategy for scheduling and processing the tasks, generated from IoT devices to suitable computing devices. The proposed strategy assigns a priority on each task based on its deadline and assigns it to a suitable multilevel-feedback queue. This schema reduces the waiting time of the delay-sensitive tasks on the queue and minimizes the starvation problem of the low priority tasks. Moreover, DPTO strategy selects an optimal computing device for each task based on its resource availability and transmission time from the IoT device. This strategy minimizes the overall offloading time of the tasks while meeting the deadlines. Finally, the extensive simulation results with various performance parameters show the effectiveness of the proposed strategy over the existing baseline algorithms.**

***Index Terms*—Fog Computing; Multilevel-feedback priority Queue; Task Offloading; Deadline; IoT**

I. Introduction

The rapid growth of Internet of Things (IoT) applications including smart city and home, smart transportation, smart grid, smart health-care, smart water and waste management have led to the emergence of a variety of delay-sensitive tasks [1]. These tasks demand a substantial computational resources for the processing in real-time environment. Past few decades, the resource constrained IoT devices offloaded the tasks to the cloud servers for processing and analyzing [2], [3]. However, the physical distance between the IoT devices and the cloud servers introduces several challenges such as latency and network congestion. To address such drawbacks, Cisco [4] introduced *fog computing* paradigm for processing delay-sensitive task offloading in local fog devices [5]–[7]. Basically, the fog devices are geographically distributed and virtually bring the cloud functionality to the edge of the network [8]–[11]. Thus, the IoT devices prefer to offload the delay-sensitive tasks to the local fog devices for minimizing the overall latency and processing time, while meeting Quality-of-Service (QoS) constraints such as deadline. However, due to limited resource capacity of the fog devices, the resource-intensive tasks are offloaded to the resource-rich cloud servers for further analysis and processing [12]. Thus, a hierarchical fog-cloud environment is more suitable for processing both the delay-sensitive and resource-intensive applications while meeting various QoS objectives [13].

Corresponding author: Satish Narayana Srirama.

M. Adhikari and S. N. Srirama are with the Mobile & Cloud Lab, Institute of Computer Science, University of Tartu, Estonia (e-mail: mainak.ism@gmail.com, satish.srirama@ut.ee).

M. Mukherjee is with the Guangdong Provincial Key Laboratory of Petrochemical Equipment Fault Diagnosis, Guangdong University of Petrochemical Technology, Maoming 525000, China (e-mail: m.mukherjee@ieee.org).

A. Motivation

In a standard hierarchical fog-cloud environment, the IoT devices offload the tasks to the suitable computing devices (*i.e.*, fog devices or cloud servers) through a gateway node. However, the IoT devices generate different types of tasks which are generally classified based on various QoS constraints such as delay deadline (*i.e.*, hard deadline or soft deadline). For a hard deadline-based task, the task should complete its processing within the assigned deadline, otherwise the output of the task is no longer valid for the IoT device. However, for soft deadline-based tasks, the output of the tasks is valid up to some extent with some penalties even though the tasks failed to meet their corresponding deadlines. Most of the existing offloading strategies in fog environment [14], [15] deploy the tasks to the nearby fog nodes or the centralized cloud servers without considering these different types of deadlines. Moreover, due to the limited computational and storage capacity of the local fog devices, the resource-insensitive tasks with no deadlines are not suitable for processing in the local fog devices. Thus, with current approaches, most of the tasks fail to meet their deadlines due to the additional burden on the latency as well as resource consumption issues of the computing devices. Therefore, the critical and yet unsolved challenge is to offload the tasks to the suitable computing devices, based on their priorities and ability to meet the deadlines and resource constraints of the tasks with minimum latency.

B. Related Work

Recently, a series of strategies and algorithms have been designed in various literature for task offloading in the field of fog-cloud environment [8], [14], [15] and mobile-edge-cloud scenarios [16]–[19]. In the fog offloading scenario, an ideal solution is to process each task on a single fog device with sufficent resource capacity [20], [21]. Yousefpour *et al.* have introduced a fog integrated cloud environment for processing delay-sensitive IoT application with minimum processing time [22]. An analytical model based on queueing theory has been designed by Souza *et al.* for minimizing the service delay and offloading time of the tasks [23]. Rodrigues *et al.* have designed a delay-sensitive task offloading strategy with virtual machine migration and transmission power control parameters for minimizing the offloading time and processing delay [24]. Fricker *et al.* have introduced an optimal offloading strategy for minimizing the offloading time while balancing the loads among the multiple fog devices [25]. He *et al.* have designed a heuristic offloading strategy for deploying the tasks on the optimal fog node in a multi-tier fog-cloud environment for minimizing the offloading time [26]. Shih *et al.* have developed a fog-radio access network for offloading ultra low-latency tasks with minimum processing time and offloading delay [27].

Yang *et al.* have designed a maximum energy efficient offloading strategy for processing intelligent IoT applications with minimum latency and processing time on a homogeneous fog environment [28]. Mansouri *et al.* have devolved an offloading game model for processing the real-time tasks on the powerful fog devices with minimum latency and computation time in a hierarchical fog-cloud environment [29]. The proposed strategy further optimized using pure Nash Equilibrium technique. Liu *et al.* have designed a hybrid computational offloading strategy in a multi-tier fog-cloud environment for minimizing the latency with efficient resource utilization [30]. Du *et al.* have devolved a computational offloading and resource allocation strategy in a mixed fog-cloud environment for minimizing the latency and energy usage of the the real-time tasks with efficient resource utilization [31]. Jiang *et al.* have designed a latency-aware offloading strategy in a multi-tier fog-cloud environment for utilizing the communication and computation resources with minimum transmission delay [32]. The existing task offloading strategies deployed the tasks to the suitable fog devices or cloud servers for minimizing the processing time and offloading delay without concerning the QoS constraint such as deadline of the tasks.

Queueing theory is being widely applied in the field of hierarchical fog-cloud environment to reduce the offloading time of the tasks to the selected computing devices [33], [34]. For example, Fan *et al.* have developed a M/M/1 queueing model considering computational delay and offloading time of the tasks in mobile-cloud environment with minimum resource cost and response time [35]. Moreover, Kumar *et al.* have designed an *on-demand* computational offloading framework using a queueing model to minimize the cost of the tasks while balancing the workload [36]. Liu *et al.* have designed an optimal offloading strategy using a linear programming method under a queueing model for minimizing the transmission delay and energy consumption of the resources [37]. Nan *et al.* have developed a queueing model with Lyapunov optimization technique in fog-cloud environment for minimizing the processing dealy and cost of the tasks while offloading to a suitable computing device [38]. Alnoman *et al.* have proposed a priority-aware scheduling strategy for delay-sensitive applications using a queueing model which minimizes the overall delay for the tasks in a fog-cloud environment [39]. Moreover, a multi-cast queueing network is suggested in fog environment for minimizing the delay and utilizing the computing resources efficiently [40].

The existing offloading strategies have used queueing model for storing the data to find an optimal computing device without considering the priority of the tasks. Basically, all the computing devices and their idle resource capacities can be selflessly process the tasks without considering their priority and constraints. However, nowadays, the IoT devices generate various types of delay-sensitive tasks which should complete to process with minimum offloading time. Thus, most of the existing strategies fail to meet the QoS constraints of the tasks with minimum offloading time.

C. *Objective and Contributions*

To overcome such issue, in this paper, we introduce a heuristic task offloading schema in a hierarchical fog-cloud environment for IoT applications using a multilevel-feedback queueing model. The main objectives of the work is to minimize the overall queueing waiting time and offloading time of the real-time tasks while meeting the deadline and resource constraints. Extensive simulation results with various performance matrices are presented to show the effectiveness of the proposed schema. The main contributions of this paper are summarized as follows.

- We consider a hierarchical fog-Cloud model for processing the tasks generated from the IoT devices. The proposed framework provisions both the ad-hoc fog nodes with distributed opportunistic computing resources and the centralized Cloud servers with sufficient computational resource capacity.
- We further developed a priority assignment strategy to classify the tasks into three categories based on their deadlines. Moreover, we designed a rule-based task scheduling strategy for finding an optimal order of the tasks and minimizing the waiting time in the queue.
- A heuristic task offloading strategy is suggested to find a suitable computing device that minimizes the overall offloading time.

The rest of the paper is organized as follows. Section II presents the system model and problem formulation of the work. The delay-dependent priority-aware task offloading strategy is discussed in Section III. The simulation results are presented in Section IV. Finally, conclusions are drawn in Section V.

II. System Model and Problem Formulation

A. *Network Model*

As illustrated in Fig. 1, we consider a hierarchical fog-Cloud scenario with a set of K IoT devices denoted as $\mathcal{K} = \{1, 2, \ldots, K\}$ and a set of locally distributed N fog devices denoted as $\mathcal{N} = \{1, 2, ..., N\}$. Furthermore, we consider a set $\mathcal{G} = \{1, 2, \ldots, G\}$ of IoT gateways that receive the tasks from the multiple IoT devices and offload these tasks to the suitable computing devices such as fog devices or remote cloud servers. Denote F_n^{MM} as the memory usage of the nth fog node.

TABLE I
MAIN NOTATION DEFINITION

Symbols	Definition
K	The number of total IoT devices in the network
N	The number of total fog nodes in the network
M	The number of total cloud servers in the data center
G	The number of total IoT gateways in the network
L	The number of total tasks
B_{ji}^{up}	The uploading bandwidth [bits/second] from ith Fog device to jth cloud server
B_{ij}^{out}	The downloading bandwidth [bits/second] from jth cloud server to ith Fog device
$\mathcal{X}_j$	Input data size [bits] of jth task
PR_i	Processing density of ith computing device
D_j	The total number of CPU cycles to process the jth task
$\mathfrak{T}_j$	The deadline of the jth task
f_i	The CPU frequency [cycles/second] of the ith computing device $i \in (\mathcal{K} \cup \mathcal{N} \cup \mathcal{M})$
f_k	The CPU frequency [cycles/second] of the kth IoT device
f_n	The CPU frequency [cycles/second] of the ith fog device
f_m	The CPU frequency [cycles/second] of the jth cloud server
S_m^{MM}	The memory usage of the mth cloud server
F_n^{MM}	The memory usage of the nth Fog device
A_j^{HD}	The jth task with hard deadline
A_j^{SD}	The jth task with soft deadline
A_j^{wout}	The jth task without any deadline

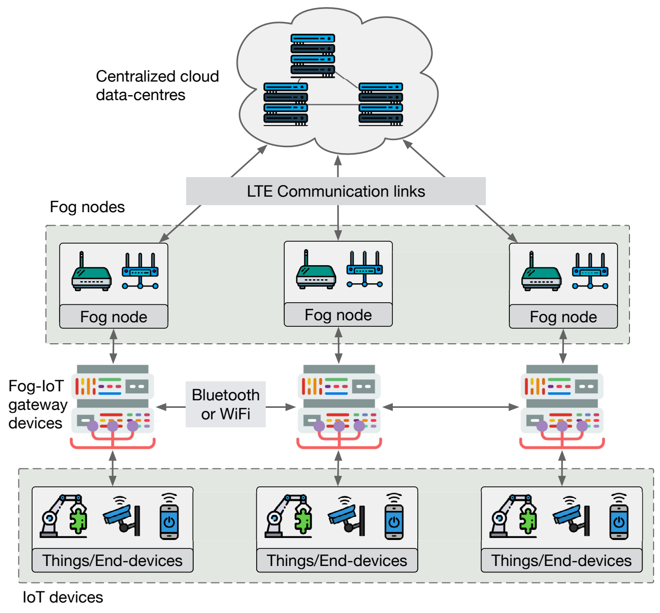


Fig. 1. Illustration of hierarchical fog-cloud model for IoT applications.

Moreover, a set of M cloud servers deployed on the centralized cloud data center is denoted as $\mathcal{M} = \{1, 2, \ldots, M\}$. Note that the CPU frequency and memory usage of the cloud servers are higher than the fog devices, *i.e.*, $f_m >> f_n$ and $S_m^{MM} >> F_n^{MM}$. Suppose that the IoT devices have L independent and identical tasks to be executed, denoted by the set $\mathcal{A} = \{1, 2, 3, \ldots, A\}$. In the hierarchical fog-cloud environment, each task is either processed locally, or offloaded to one of the N fog devices for local execution or M cloud servers for remote execution. Denote $\pi \in \mathbb{R}^{L \times |\mathcal{K} \cup \mathcal{N} \cup \mathcal{M}|}$ as the task assignment matrix, where the (j, i)th entry is represented as $\pi(j, i) \in \{0, 1\}, j \in L, i \in (\mathcal{K} \cup \mathcal{N} \cup \mathcal{M})$, is given by

$$\pi(j,i) = \begin{cases} 1 & \text{if the } j\text{th task is assigned to the } i\text{th computing device,} \\ 0 & \text{otherwise.} \end{cases}$$

Ler $A_L = \{j \in L : \pi(j, i) = 1\}$ denotes the set of tasks which are assigned to computing device $i, i \in (\mathcal{K} \cup \mathcal{N} \cup \mathcal{M})$. We also assume that $|\pi(j, i)| \geq 1$, *i.e.*, each task should be assigned to at least one computing device. The total number of CPU cycles required to process jth task is expressed as $D_j = \mathcal{X}_j \times PR$, where $\mathcal{X}_j$ is the input data size of the task and PR is the processing density of the task. The main notations in this paper are summarized in Table I.

1) Local Computing: The tasks that require less computing resources for processing and storage are computed locally at the IoT device. The processing time of the jth task in the ith computing device is expressed as

$$T_{ji} = \frac{\sum_{j=1}^{L} \pi(j,i)\, D_j}{f_i}, \ \forall i \in (\mathcal{N} \cup \mathcal{M}), j \in \mathcal{K}\,. \quad (1)$$

Note that if the task is locally computed at the IoT device, then the total computation mainly depends on the CPU frequency of the IoT device instead of the communication delay.

2) Remote Computing: Due to the limited processing and storage capacity of the IoT devices, most of the tasks are uploaded to the either locally distributed fog devices or centralized cloud servers. The deadline of the task and communication delay between the IoT device and computing device (fog or cloud server) will play a major role for the offloading decision. We discuss the phases as follows.

Phase 1 - Task Uploading: At first, the tasks are uploaded to the suitable computing devices via the local IoT gateway. For simplicity, we assume that the tasks are uploaded to the local fog node or the centralized cloud servers based on their priority and the deadline. Let us consider, hp_j denotes the channel power gain from the local IoT devices for offloading the jth to the ith computing device, $i \in (\mathcal{N} \cup \mathcal{M})$. The achievable uploading rate (in bps) at the ith computing device is defined as follows.

$$R_{ji}^{\text{up}} = B_{ji}^{\text{in}} \log_2 \left(1 + \frac{P_i^{\text{up}} hp_j}{\xi_i^2}\right), \forall i \in (\mathcal{N} \cup \mathcal{M}), j \in \mathcal{K} \quad (2)$$

where B_{ji}^{in} represents the available transmission bandwidth between jth IoT device to ith computing device; P_i^{up} denotes the transmission power for uploading to the ith computing device; and ξ_i^2 represents the additive white Gaussian noise of the ith computing device.

The task uploading time to the ith computing device is defined as follow.

$$T_{ji}^{\text{up}} = \frac{\sum_{j=1}^{L} \pi(j,i) \mathcal{X}_j}{R_{ji}^{\text{up}}}, \forall i \in (\mathcal{N} \cup \mathcal{M}), j \in \mathcal{K} \quad (3)$$

Phase 2 - Task Processing: After receiving the assigned task to the ith computing device, the task process with the computational frequency of that device, *i.e.*, $f_i : i \in (\mathcal{N} \cup \mathcal{M})$. The total processing time of the jth task is expressed as

$$T_{ji} = \frac{\sum_{j=1}^{L} \pi(j,i)\, D_j}{f_i}, \forall i \in (\mathcal{N} \cup \mathcal{M}), j \in \mathcal{K}\,. \quad (4)$$

Phase 3 - Task Downloading: After processing the assigned task, the ith computing device begins to transmit the task to the requested IoT device. Similar to the Task uploading phase, the computing devices transmit their results to the respective IoT devices based on their completion. Let gp_i denotes the channel power gain from the ith computing device for downloading the result to the respective jth IoT devices. The achievable downloading rate from the ith computing device is defined as follow.

$$R_{ij}^{\text{down}} = B_{ij}^{\text{out}} \log_2 \left(1 + \frac{P_i^{\text{down}} gp_i}{\xi_j^2}\right) \forall i \in (\mathcal{N} \cup \mathcal{M}), \forall j \in \mathcal{K} \quad (5)$$

where B_{ij}^{down} represents the available transmission bandwidth between ith computing device and jth IoT device and $B_{ij}^{\text{out}} = B_{ji}^{\text{in}}$; P_i^{down} denotes the transmission power for downloading the ith computing device; and ξ_j^2 represents the additive white Gaussian noise of the jth IoT device. The task downloading time to the ith computing device is expressed as

$$T_{ij}^{\text{down}} = \frac{\sum_{j=1}^{L} \pi(j,i) \mathcal{X}_j}{R_{ij}^{\text{down}}}, \forall i \in (\mathcal{N} \cup \mathcal{M}), j \in \mathcal{K}\,. \quad (6)$$

3) Total Offloading Time: The IoT-based tasks are either processed locally or should be offloaded to the local distributed fog devices or the centralized cloud servers based on their deadline and resource availability of the computing devices. For local processing, the total offloading time of the tasks only depends on their processing time on the IoT devices, *i.e.*, $T_{j,i}^{\text{offload}} = T_{ji}$ (see, (1)). However, the total offloading time of the IoT-based tasks for remote computing depends on the transmission time of the data uploading and downloading and the processing time on the selected computing devices. Thus, the total offloading time of the jth task on the ith computing device is defined as follows.

$$T_{j,i}^{\text{offload}} = \begin{cases} T_{ji} & \forall i \in A_L, j \in \mathcal{K} \\ T_{ji}^{\text{up}} + T_{ji} + T_{ij}^{\text{down}} & \forall i \in (\mathcal{N} \cup \mathcal{M}), j \in \mathcal{K}\,. \end{cases} \quad (7)$$

B. Queueing Model

As illustrated in Fig. 2, the IoT gateways receive multiple number of tasks from different IoT devices. Here, we consider a time-slotted system, $t = \{1, 2, 3, ..., t\}$, where the length of each time-slot is Δt. Each IoT gateway is idle and all the queues are empty when $t < 0$. The global queue of each IoT gateway follows the M/M/I – First-Come-First-Serve (FCFS)/$\mathcal{Q}$-bounded queueing model with a birth-death process where $I\, \forall I \in (\mathcal{N} \cup \mathcal{M})$ denotes the number of active computing devices as a form of fog devices or cloud servers and $\mathcal{Q}$ represents the size of the queue. Here, we consider the occurrence of a sequence of discrete tasks can be realistically modeled as a Poisson process, *i.e.*, the time interval between the arrival of the successive tasks are exponentially distributed. For a Poisson process, the time interval between two such arrivals of tasks are treated as independent random variables which are drawn from an exponential distributed population with the density function $f(x) = \lambda e^{-\lambda x}$ for some fixed constant λ, which is also called as the arrival rate of the tasks. Denote μ as the service rate of the computing devices. We assume same service rate for all the computing devices. Therefore, the utilization of each computing device is defined as $\rho = \lambda Z_s / \mu$, where Z_s denotes the number of active computing devices, where, $s \in (\mathcal{N} \cup \mathcal{M})$. The probability that all computing devices are idle at certain time-stamp is defined as follows.

$$P_0 = \frac{1}{\sum_{i=1}^{Z_s} \left(\frac{(\lambda^L/\mu^L)}{L!} + \frac{(\lambda^{Z_s}/\mu^{Z_s})}{Z_s!} \frac{1}{1 - (\lambda / Z_s\, \mu)} \right)} \quad (8)$$

The probability that the ith computing device is in an *idle*-state at a certain time-stamp is defined as follows.

$$P_i = \begin{cases} \frac{\lambda^i/\mu^i}{i!} P_0 & \text{where } i = 1, 2, \ldots, Z_s \\ \frac{\lambda^i/\mu^i}{Z_s!\, Z_s^{(i-Z_s)}} P_0 & \text{where } i = Z_s + 1, Z_s + 2, \ldots \end{cases} \quad (9)$$

Moreover, the mean number of tasks waiting at the queue of the IoT gateway is expressed as

$$\bar{Q} = \frac{P_0 \left(\lambda^{Z_s}/\mu^{Z_s}\right) \rho}{Z_s!\, (1-\rho)^2}\,. \quad (10)$$

Afterward, the IoT gateway classifies the tasks into three categories based on their deadline: hard deadline (A_j^{HD}), soft deadline (A_j^{SD}), and no deadline (A_j^{wout}) tasks. These tasks are placed in three *multilevel-feedback* priority queues denoted as $\mathbb{Q}_x$, where $x = 1, 2, 3$, as shown in Fig. 2. The tasks with hard deadlines have highest priority and place them on the high priority multilevel-feedback queue $\mathbb{Q}_1$, however, the tasks with no deadline have lowest priority and place them on low priority multilevel-feedback queue $(\mathbb{Q}_3)$. To avoid the starvation problem of the soft deadline or no deadline tasks, here we consider the multilevel-feedback queue. After a certain time-stamp, if the lower priority tasks are not assigned to any computing device for processing, then the priority of the tasks are increased by 1 and the tasks are promoted to higher priority queue. Thus, the total waiting time of a task j on the priority queue $(\mathbb{Q}_x)$ is defined as follows.

$$T_j^{\text{wait}} = \frac{\lambda_j Z_s^2}{2(1-\rho_j)} = \frac{Z_s^2}{\mu_j (1-\rho_j)}\,. \quad (11)$$

C. Problem Formulation

The main objective is to minimize the offloading delay (uploading time, remote task processing, and downloading time) and waiting time in the queue for local or remote computing for all the tasks coming from the IoT devices. The main intention of this work is to schedule the tasks optimally and assign most of the hard-deadline (A_j^{HD}) and soft-deadline (A_j^{SD}) based tasks to the local fog device $i, \forall i \in$

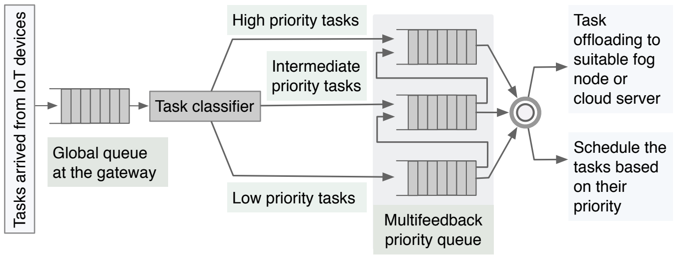

Tasks arrived from IoT devices
Global queue at the gateway
Task classifier
High priority tasks
Intermediate priority tasks
Low priority tasks
Multifeedback priority queue
Task offloading to suitable fog node or cloud server
Schedule the tasks based on their priority

Fig. 2. Multilevel-feedback priority queueing model for IoT gateway.

$\mathcal{N}$, which minimizes the latency and overall offloading time. However, the tasks without deadline (A_j^{wout}) can be deployed on the centralized cloud server $i, \forall i \in \mathcal{M}$ for optimizing the performance of the system model. Thus, the aim of this work is to optimize the task assignment strategy for the jth task on the ith computing device, *i.e.*, $\pi(j,i)$, the task uploading time T_{ji}^{up}, processing time T_{ji}, and downloading time T_{ij}^{down} with the deadline constraint A_j^D. As a result, the proposed strategy minimizes the overall queueing waiting time T_j^{wait} and total offloading time T_{ji}^{offload} of the set of tasks $A_j, \forall A_j \in (A_j^{\text{HD}} \cup A_j^{\text{SD}} \cup A_j^{\text{wait}})$.

We formulate the above objective and constraints as follows:

$$\text{minimize} \quad T_{ji}^{\text{offload}} + T_j^{\text{wait}} \quad \forall j \in \mathcal{K}, i \in (\mathcal{K} \cup \mathcal{N} \cup \mathcal{M}) \tag{12a}$$

$$\text{subject to} \quad T_{ji}^{\text{offload}} + T_j^{\text{wait}} \leq A_j^D, \forall A_j^D \in (A_j^{\text{HD}} \cup A_j^{\text{SD}}), \tag{12b}$$

$$A_j^{CPU} \leq R_i^{CPU}, R_i^{CPU} \in (F_i^{CPU} \cup S_i^{CPU}), \tag{12c}$$

$$A_j^{MM} \leq R_i^{MM}, R_i^{MM} \in (F_i^{MM} \cup S_i^{MM}), \tag{12d}$$

$$\pi(j,i) \in \{0,1\}, \tag{12e}$$

$$\sum_{i=1}^{Z_s} \pi(j,i) = 1, \tag{12f}$$

$$T_{ji}^{\text{up}} \geq 0, \text{ and } T_{ij}^{\text{down}} \geq 0 \tag{12g}$$

In the above problem, the objective function is to minimize the total offloading time and queueing waiting time of the tasks which is addressed in 12. The constraint 12b states that the total completion time of the tasks should meet their corresponding deadline. The constraint 12c and 12d state that the CPU frequency and memory usage of the tasks should be less than or equal to the selected computing devices $i, \forall i \in (\mathcal{N} \cup \mathcal{M})$ respectively. Constraint 12e imposes the binary offloading constraint. As a result, a task A_j should complete its processing on the assigned computing device $i, \forall i \in (\mathcal{N} \cup \mathcal{M})$. Constraint 12f represents that each task must be assigned to a computing device $i, \forall i \in (\mathcal{N} \cup \mathcal{M})$. Finally, the constraint 12g represents that the task offloading time, i.e., T_{ji}^{up} and task downloading time, i.e., T_{ij}^{down} should be non-negative.

III. Proposed Delay-Dependent Priority-Aware Task Offloading (DPTO)

In this section, we detail the description of the delay-dependent priority-aware task offloading (DPTO) strategy for the tasks in a fog-cloud environment. The primary goal of the strategy is to schedule the tasks based on their priority and offload them to the suitable computing devices for minimizing the delay while meeting the deadlines. Our proposed DPTO strategy has three phases as follows. a) In first phase, the IoT gateway verifies each task based on its deadlines and assigns a priority. b) In the next phase, the task scheduler finds an optimal scheduling strategy for minimizing the waiting time of the tasks in queue and optimizes the starvation problem of the low priority tasks. c) Finally, the scheduled tasks are offloaded to a suitable computing device that minimizes the total latency while meeting the deadlines of the tasks. The details of the above-mentioned three phases of DPTO strategy are discussed as follows:

A. Priority Assignment

In this phase, based on the predefined deadline, the task classifier (shown in Fig. 2) assigns the priority of each task waiting in the global queue of each IoT gateway. Specifically, we obtain three categories as follows:

- **Task Priority 1** ($\mathbb{A}_1$). It is the highest priority class that aims to support the delay-sensitive tasks with hard deadlines. Such type of tasks should meet their deadline without any negotiation, *i.e.*, the tasks will be killed if they did not meet their respective deadline. This type of tasks are either processed locally in the IoT devices or assigned to the local fog devices in order to avoid significant offloading and downloading time in case of remote cloud. Specifically, a more reliable computing devices are assigned for such tasks.
- **Task Priority 2** ($\mathbb{A}_2$). This type of tasks have intermediate priority with *soft*-deadline. Basically, such type of tasks meet their deadline with a negotiation of total latency, *i.e.*, additional time may be provided for completion of the task processing. This type of tasks can be either locally processed at the fog devices or offloaded to the centralized cloud servers.
- **Task Priority 3** ($\mathbb{A}_3$). It is the lowest priority class that aims to support the resource-intensive tasks without any deadline. Such type of tasks require more resource capacity (*i.e.*, CPU frequency and memory) for processing without a primary constraint on minimizing the latency. This type of tasks are mostly offloaded to the centralized cloud servers which have sufficient processing and storage resources.

We should emphasise that all tasks with priority $\mathbb{A}_1$ must be sent to the highest priority queue (denoted as $\mathbb{Q}_1$) in the multilevel-feedback queue, thus, $\mathbb{A}_1$ is assigned to $\mathbb{Q}_1$. Moreover, the tasks with priority $\mathbb{A}_2$ are assigned to the intermediate priority multilevel-feedback queue denoted as $\mathbb{Q}_2$. Besides, the low priority multilevel-feedback queue $\mathbb{Q}_3$ receives the tasks with low priority tasks $\mathbb{A}_3$.

B. Task Scheduling

This phase finds an optimal scheduling order of the tasks for offloading on the computing devices. A task $A_j, \forall (j \in A_j^{\text{HD}} \cup A_j^{\text{SD}} \cup A_j^{\text{wout}})$ is scheduled to the IoT Gateway for offloading based on the following rules.

Rule 1: If $\mathbb{Q}_1 \neq \varnothing$, then the tasks with priority $\mathbb{A}_1$ are scheduled in FCFS order by the IoT gateway for offloading.

Rule 2: If $\mathbb{Q}_1 = \varnothing$ and $\mathbb{Q}_2 \neq \varnothing$, then the tasks with priority $\mathbb{A}_2$ are scheduled in FCFS order by the IoT gateway for offloading.

Rule 3: If $\mathbb{Q}_1 \neq \varnothing$ and $\mathbb{Q}_2 \neq \varnothing$ and $T_j^{\text{wait}} \geq \eta_1, \forall j \in A_j^{\text{SD}}$, then the tasks with priority $\mathbb{A}_2$ are promoted and placed at the end of the higher priority queue $\mathbb{Q}_1$ for further scheduling.

Rule 4: If $\mathbb{Q}_1 = \varnothing$ and $\mathbb{Q}_2 = \varnothing$ and $\mathbb{Q}_3 \neq \varnothing$, then the tasks with priority $\mathbb{A}_3$ are scheduled in FCFS order by the IoT gateway for offloading.

Rule 5: If $\mathbb{Q}_1 \neq \varnothing$ and $\mathbb{Q}_2 \neq \varnothing$ and $\mathbb{Q}_3 \neq \varnothing$ and $T_j^{\text{wait}} \geq \eta_2, \forall j \in A_j^{\text{wout}}$, then the tasks with priority $\mathbb{A}_3$ are promoted and placed at the end of the intermediate priority queue $\mathbb{Q}_2$ for further scheduling.

Rule 6: If $\mathbb{Q}_1 \neq \varnothing$ and $\mathbb{Q}_2 = \varnothing$ and $\mathbb{Q}_3 \neq \varnothing$ and $T_j^{\text{wait}} \geq \eta_2, \forall j \in A_j^{\text{wout}}$, then the tasks with priority $\mathbb{A}_3$ are promoted and placed at the end of the higher priority queue $\mathbb{Q}_1$ for further scheduling.

Rule 7: If $\mathbb{Q}_1 = \varnothing$ and $\mathbb{Q}_2 = \varnothing$ and $\mathbb{Q}_3 = \varnothing$, then the overall process should be suspended.

Here, we consider two types of waiting time constraints for intermediate priority tasks and low priority tasks in the queue, such as η_1 and η_2 for avoiding starvation problem. According to Rule 3, if the higher priority queue $\mathbb{Q}_1$ is not empty, *i.e.*, $\mathbb{Q}_1 \neq \varnothing$ and few tasks waiting time greater than threshold value η_1, *i.e.*, $T_j^{\text{wait}} \geq \eta_1, \forall j \in A_j^{\text{SD}}$ of the intermediate priority queue, then the priority of such tasks are promoted to priority $\mathbb{A}_1$. Similarly, according to Rule 5, if the higher priority queue $\mathbb{Q}_1$ and intermediate priority queue $\mathbb{Q}_2$ are not empty ($\mathbb{Q}_1 \neq \varnothing$) and few tasks waiting time greater than threshold value η_2, *i.e.*, $(T_j^{\text{wait}} \geq \eta_2), \forall j \in A_j^{\text{wout}}$ of the no priority queue, then the priority of such tasks are promoted to priority $\mathbb{A}_2$, which is also applicable for Rule 6 as well. The formulation of the waiting time of a task T_j^{wait} is defined as follows.

At first, the waiting time of a task A_j with hard deadline (A_j^{HD}) in higher priority queue $\mathbb{Q}_1$ depends on the previous tasks present in the queue $\mathbb{Q}_1$ which is calculated as follows.

$$T_j^{\text{wait,HD}} = \begin{cases} 0 & \text{where } \mathsf{pred}(\mathbb{Q}_1 = \varnothing) \\ \sum_{a=1}^{N_1} T_a^{\text{wait}} & \forall a \neq j, A_a \in \mathsf{pred}(A_j^{\text{HD}}) \end{cases} \tag{13}$$

Here, N_1 is the length of $\mathbb{Q}_1$, and $\mathsf{pred}(.)$ denotes the previous set of tasks arrived in a queue. Secondly, the waiting time of a task A_j with soft deadline (A_j^{SD}) in intermediate priority queue $\mathbb{Q}_2$ depends on the tasks available in the higher priority queue $\mathbb{Q}_1$, which is defined as follows.

$$T_j^{\text{wait,SD}} = \begin{cases} T_j^{\text{wait,HD}} & \text{where } A_j \in A_j^{\text{HD}} \\ T_j^{\text{wait,HD}} + \sum\limits_{b=1}^{N_2} T_b^{\text{wait}} & \forall b \neq j, A_b \in \mathsf{pred}(A_j^{\text{SD}}) \end{cases} \tag{14}$$

where N_2 is the length of $\mathbb{Q}_2$. Finally, the waiting time of a task A_j without deadline (A_j^{wout}) in low priority queue $\mathbb{Q}_3$ depends on the tasks available in the higher priority queue $\mathbb{Q}_1$ and intermediate priority queue, which is calculated as follows.

$$T_j^{\text{wait, wout}} = \begin{cases} T_j^{\text{wait,SD}} & \text{where } A_j \in A_j^{\text{SD}} \\ T_j^{\text{wait,SD}} + \sum\limits_{c=1}^{N_3} T_c^{\text{wait}} & \forall c \neq j, A_c \in \mathsf{pred}(A_j^{\text{wout}}) \end{cases} \tag{15}$$

where N_3 denotes the length of $\mathbb{Q}_3$.

C. Task Offloading

The task offloading strategy uses a heuristic policy for finding a suitable computing device for each scheduled task that minimizes overall offloading time while meeting the deadline. Here, we assume that each IoT gateway $\mathcal{G}$ connects with multiple fog devices and cloud servers, *i.e.*, $\mathcal{G} \rightarrow i, \forall i \in (N, M)$. However, it may also be possible that a fog device $i, \forall i \in N$ is connected with multiple IoT gateways, *i.e.*, $i \rightarrow (\mathcal{G}_x, \mathcal{G}_y), \forall i \in N$, where $x \neq y$. So, each gateway should communicate with the connected fog devices for collecting their current status (*i.e.*, busy or free) and resource availability in the form of CPU frequency and memory usage. Moreover, we also assume that the fog devices and cloud servers are placed in a static environment, *i.e.*, the distance between the computing devices and IoT gateways with communication bandwidth are known as prior. So, the uploading time (T_{ki}^{up}) and downloading time $(T_{ik}^{\text{down}}), \forall i \in (\mathcal{N} \cup \mathcal{M}), k \in \mathcal{G}$ between ith computing device and kth gateway are calculated using (3) and (6) respectively.

Now, the proposed heuristic finds a suitable computing device i for each scheduled task A_j which requires minimum uploading and downloading time and meets the resource requirements of the tasks. The IoT gateway should deploy the deadline based tasks A_j^D, where $A_j^D \in (A_j^{\text{HD}} \cup A_j^{\text{SD}})$ to the local fog nodes that have sufficient CPU frequency for processing the tasks within their deadline $\mathfrak{T}_j$. However, the tasks without deadlines $A_j \in A_j^{\text{wout}}$ should be offloaded to a suitable computing device that meets their resource requirements. The required CPU frequency for processing task A_j^D in a computing device i is defined as

$$Rf_j^C = \frac{\mathcal{X}_j}{\mathfrak{T}_j - \left(T_j^{\text{wait}} + AU_{ki}^{\text{up}} + Ad_{ik}^{\text{down}} + RRT_{ki}\right)}, \tag{16}$$

where RRT_{ki} represents the round-trip time between the computing device $i, \forall i \in (\mathcal{N} \cup \mathcal{M})$ and IoT gateway $k, \forall k \in G$ for finding whether the computing device i is free to process the upcoming task A_j or busy for processing another task A_j, coming from other gateway device k. Let us consider that the required memory usage of each task A_j for storing output data is R_j^{MM} and the available memory usage of a computing device i is represented as CD_i^{MM}. The selection of a computing device i for each scheduled deadline based tasks A_j^D is based on the following rules.

- *Rule 1:* If $Rf_j^C \leq f_i^C$ and $R_j^{MM} \leq CD_i^{MM} \forall j \in A_j^D, k \in (\mathcal{N} \cup \mathcal{M})$, then deploy the task A_j^D on the computing device i. Here, the deadline based tasks are

Algorithm 1: Proposed DPTO Strategy

```
input : Tasks with deadlines, $Z_j, D_j, T_j^{\text{wait}}, T_{ji}^{\text{up}}, T_{ij}^{\text{down}}$
output: Suitable $i$th computing device for the task
1  begin
2      for $j = 1$ to $l$ do
3          Assign the priority of tasks using Rule 1 to Rule 7 of Task
             Scheduling phase:;
4          Waiting Time of tasks are calculated using (13)–(15);
       end
5      for $i = 1$ to $Z_s : \forall i \in (\mathcal{N} \cup \mathcal{M})$ do
6          for $j = 1$ to $l$: $\forall j \in A_j^D$ do
7              Calculate $Rf_j^C$ using (16);
8              if $Rf_j^C \leq f_i^C$ and $R_j^{MM} \leq CD_i^{MM}$ then
                   Deploy task $A_j^D$ on computing device $i$
               end
9              if $Rf_j^C \geq f_i^C$ and $R_j^{MM} \leq CD_i^{MM}$ then
                   Not Possible to deploy
               end
10             if $Rf_j^C \leq f_i^C$ and $R_j^{MM} \geq CD_i^{MM}$ then
                   Not Possible to deploy
               end
11             if $Rf_j^C \geq f_i^C$ and $R_j^{MM} \geq CD_i^{MM}$ then
                   Not Possible to deploy
               end
           end
       end
   end
```

offloaded to the computing device i that meets the required CPU frequency and memory usage of the task A_j^D.

- *Rule 2:* If $Rf_j^C \geq f_i^C$ and $R_j^{MM} \leq CD_i^{MM} \forall j \in A_j^D, k \in (\mathcal{N} \cup \mathcal{M})$, then the task A_j should not be deployed on the computing device i due to lack of CPU frequency.
- *Rule 3:* If $Rf_j^C \leq f_i^C$ and $R_j^{MM} \geq CD_i^{MM} \forall j \in A_j^D, k \in (\mathcal{N} \cup \mathcal{M})$, then deploy the task A_j^D on the computing device i due to lack of memory usage.
- *Rule 4:* If $Rf_j^C \geq f_i^C$ and $R_j^{MM} \geq CD_i^{MM} \forall j \in A_j^D, k \in (\mathcal{N} \cup \mathcal{M})$, then deploy the task A_j^D on the computing device i due to lack of CPU frequency and memory usage.

The tasks without deadlines $A_j, \forall j \in A_j^{\text{wout}}$ requested a CPU frequency ($Rf_j^C, \forall j \in A_j^{\text{wout}}$) and memory usage ($R_j^{MM}, \forall j \in A_j^{\text{wout}}$) for processing. Those tasks deploy on the suitable computing device that should meet the resource requirements using *Rule 1* to *Rule 4* of the deadline-based tasks. The main goal of this phase is to minimizes the overall offloading time of the tasks while meeting their resource requirements. The pseudo code of the proposed DPTO strategy is shown in Algorithm 1.

IV. Empirical Evaluation

In this section, we evaluate and compare the performance of the proposed DPTO strategy with the following baseline algorithms in terms of a) *average queueing waiting time*, b) *average offloading delay*, and c) *the number of tasks satisfying the delay deadline* as:

- *Random offloading*: In this offloading, every IoT device *randomly* selects a computing device to offload its task.
- *Highest Data-Rate (HDR) offloading*: Every IoT device offloads its task to the computing device with minimum uploading time.
- *Highest Computing Device (HCD) offloading*: The IoT device offloads the tasks to the computing device that has sufficient CPU frequency for processing the tasks.

Basically, these baseline algorithms act as reference to show the performance improvement of the proposed offloading strategy in the fog-cloud environment.

TABLE II
Simulation Parameters

Parameters	Values
Number of fog devices (N)	10
Number of IoT devices (K)	20
Number of cloud servers (M)	2
Number of IoT gateways (G)	3
Number of tasks (L)	100
Maximum Bandwidth (BW)	20 MHz
CPU frequency of the IoT devices (f_k)	600×10^6 [cycles/s]
CPU frequency of the fog devices (f_n)	5×10^9 [cycles/s]
CPU frequency of the cloud servers (f_m)	10×10^9 [cycles/s]
Memory capacity of the IoT devices (A_k^{MM})	128 MB
Memory capacity of the fog node (F_n^{MM})	512 MB
Memory capacity of the cloud servers (S_m^{MM})	64 GB

A. Simulation Setup

For the simulations, we take 20 IoT devices that generate multiple tasks in a time interval Δt. The input and output data sizes of each task are assumed to be uniformly distributed from 10 MB to 30 MB and from 1 MB to 30 MB, respectively. The maximum transmission bandwidth between the IoT devices and the remote fog and cloud servers are 20 Mbps with no additional limit on the data uploading or downloading. For the real-time tasks, to take the deadline of the tasks into consideration, three different types of tasks are considered, hard-deadline based tasks, soft-deadline based tasks and without deadline based tasks. An empirical test of each parameter is evaluated with 1000 independent runs for finding optimal solution of each parameter. Other default simulation parameters are summarized in Table II.

B. Average Queueing Waiting Time

This metric answers the question that how much time the tasks should wait in the queue before offloading to the suitable computing devices for processing. Therefore, it can not only reflect the performance of individual task, but also minimizes the waiting time of the tasks in the queue. Fig. 3(a) illustrates the queueing waiting time of the tasks with various deadlines, generated from the IoT devices. As illustrated in Fig. 3(a), the queueing waiting time of the hard deadline-based tasks are minimum rather than the soft deadline-based and no deadline-based tasks. This is because, the hard deadline-based tasks have higher priority and need to process faster than the other tasks. Similarly, the queueing waiting time of the no deadline-based tasks are lower than the no deadline-based tasks due to their second highest priority for processing.

Fig. 3(b) shows the performance comparison of the average queueing waiting time of the tasks with the baseline algorithms. The proposed DPTO strategy schedules the hard-deadline and soft-deadline based tasks immediately for offloading instead of the without deadline based tasks. In general,

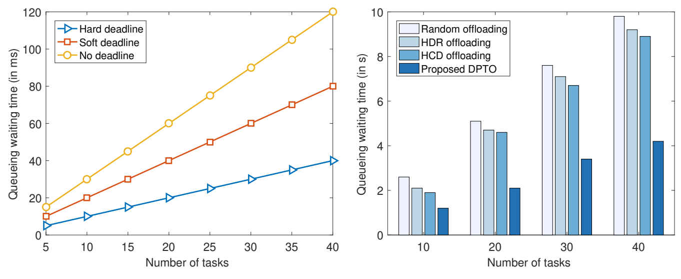


Fig. 3. Performance of (a) Queueing waiting time and (b) average queueing waiting time.

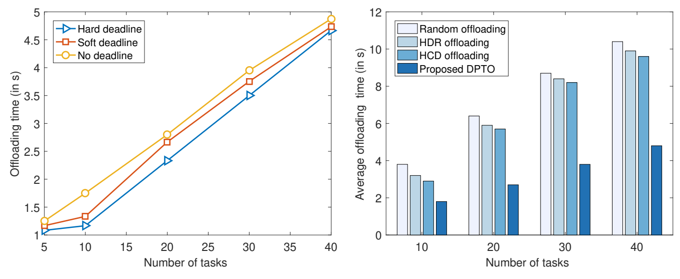


Fig. 4. Performance of (a) offloading time and (b) average offloading time

the IoT devices generate the deadline-based (either *hard* or *soft*) tasks. As a result, the proposed strategy minimizes the waiting time of each task in the queue which also reflects the overall queueing waiting time. Thus, overall queueing waiting time is minimized with increasing the number of tasks in the queue. This phenomena can also reflect the performance of the offloading time of the tasks. While the baseline algorithms deploy the tasks in FCFS order without considering their priorities and deadlines. This increases the waiting time of the deadline-based tasks, thereafter has an adverse impact on the performance of the offloading time. Thus, from the simulation results, it is observed that the DPTO strategy performs better than the baseline algorithms.

C. Average Offloading Time

The minimum queueing waiting time of the higher priority tasks also reduces the total offloading time. This metric shows the total completion time of the tasks including the uploading time, processing time and downloading time. Therefore, it not only reflects the individual-level offloading time performance, but also depicts the satisfactory level of the higher priority tasks rater than lower priority tasks. Fig. 4(a) illustrates the offloading time of the tasks with different deadlines. As illustrated in Fig. 4(a), the task offloading times increased as the number of tasks increases. This is because that the limited computing capabilities and communication bandwidth of the fog devices are insufficient when the IoT devices generate more number of tasks with deadlines for processing. Moreover, Fig. 4(a) shows that the offloading delay of the higher-priority tasks are lower than the lower-priority tasks, which improves the performance of the overall system.

Fig. 4(b) depicts the tradeoff between the proposed DPTO strategy with the baseline algorithms in term of offloading time. The DPTO strategy deploys the tasks to the suitable computing devices based on the minimum communication overhead and the availability of the computing resources for processing the tasks. While the HDR offloading algorithm deploys the tasks to the set of computing devices with minimum transmission time without considering the resource availability, however, the HCD offloading algorithm considers the resource availability without considering the transmission delay. Besides, it can be observed that the RT offloading strategy selects the computing devices randomly without considering any objective parameters. Thus, it is observed that the average offloading time of the tasks using existing baseline algorithms increases gradually rather than proposed DPTO algorithm as the number of tasks increases. Thus, the proposed DPTO strategy generates better results and minimizes the average offloading time as compared with the baseline algorithms.

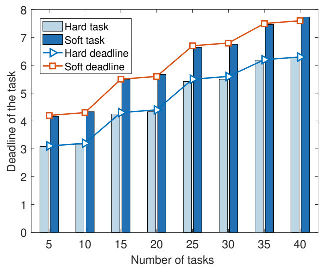
D. Number of Tasks Satisfying the Delay Deadline

This metric represents the number of tasks (with hard- and soft-deadline) satisfying their deadline while processing on various computing devices. Particularly, the occurrence of $T_{ji}^{\text{offload}} + T_j^{\text{wait}} \leq \mathfrak{T}_j \ \forall A_j^D \in (A_j^{HD} \cup A_j^{SD})$ mainly depends on the queueing waiting time and offloading time. Fig. 5(a) illustrates the number of tasks with different categories meeting their different deadlines. From Fig. 5(a), it is clear that the tasks with hard-deadlines mostly meet their deadline, however, in some scenario, the soft-deadline tasks fail to meet their deadline due to the limited resource constraint and communication overhead of the local fog nodes. Thus, for most of the cases, the proposed DPTO strategy satisfies the requirements of the tasks generated from the IoT devices.

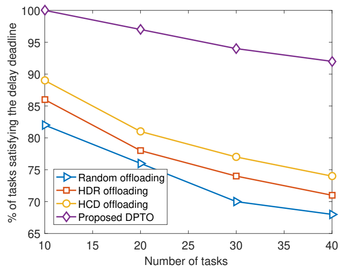
Moreover, Fig. 5(b) presents the comparative performance of the proposed DPTO strategy with the baseline algorithms in terms of number of tasks meeting the deadline. Here, we consider four different scenarios while varying different number of tasks. From Fig. 5(b), it can be clearly observed that the baseline algorithms fail to meet the deadlines as the number of tasks increase in the queue. On contrary, the proposed DPTO strategy first allocates the hard-deadline tasks followed by the soft- and no-deadline tasks. This improves the performance of the network and further increases the number of tasks that meet their deadline.

E. Performance of Throughput

The throughput of the tasks depends on the number of tasks that complete their processing within a certain time-stamp. Here, Fig. 6(a) represents the total number of tasks completing their processing within a given time-stamp. From Fig. 6(a), it is clear that most of the cases the hard deadline-based tasks complete their operation rather than the soft deadline and no deadline tasks. The DPTO strategy deploys the deadline-based tasks to the local fog nodes which complete the processing of the tasks with minimum communication overhead and

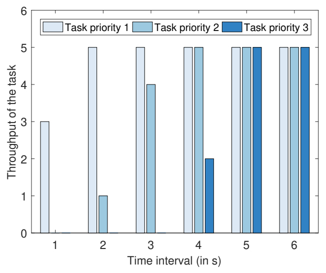


(a)

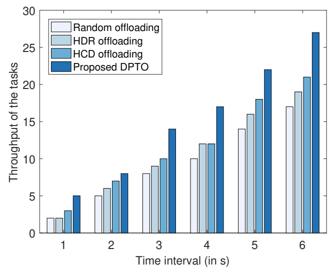


(b)

Fig. 5. (a) Number of different types of tasks meeting the deadlines and (b) average number of tasks meeting deadlines.



(a)



(b)

Fig. 6. (a) Throughput of different deadline-based tasks and (b) average throughput.

increases the throughput of the tasks. This parameter proves the efficiency of the proposed DPTO strategy.

However, Fig. 6(b) represents the comparative analysis of the proposed DPTO strategy with the existing baseline algorithms in term of throughput. Here, we calculate the number of tasks completing their processing based on the arrival rate within certain time-stamp. From Fig. 6(b) it is clear that the existing strategies fail to process more number of tasks within a time-stamp. However, the proposed strategy deploys the tasks to the local fog devices which meet the resource requirements of the tasks with minimum offloading time. This minimizes the total processing of the tasks and also increases throughput. Despite that, our scheme can roughly increase the throughput as 32% - 36%, 28% - 32% and 29% - 35% than random offloading, HDR offloading and HCD offloading algorithm, respectively.

V. Conclusion

In this paper, we investigate the task offloading strategy in a hierarchical fog-cloud environment with a multilevel-feedback queueing model. The key idea of this work is to minimize the total offloading time of the tasks while meeting their deadlines. Moreover, our proposed strategy considers the priority of the tasks based on their deadlines and assigns them on different priority queues for minimizing the waiting time of the higher priority tasks. In addition, this strategy reduces the starvation problem of the low priority tasks using multilevel-feedback queueing model. The simulation results suggested that the effectiveness of the proposed DPTO strategy over the standard baseline algorithms with numerous parameter settings. In addition, the simulation results indicate that both the minimum queueing waiting time and offloading time have positive influence to meet the deadline of the tasks based on priority-aware scheduling strategy. Our future work includes the task preemption strategy in the fog devices for improving the performance of the offloading strategy, afterward reducing the waiting time of higher priority tasks. In addition, it is worthwhile to investigate the performance of proposed offloading strategy under different resource configuration while scaling up the number of IoT devices. Moreover, as a part of future work, we plan to study the applicability of deploying the containers and serverless frameworks on the fog devices for better utilization of the computing devices while meeting the high-reliability constraint.

Acknowledgement

This work has been partially supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825040 (RADON).

References

[1] P. Schulz, M. Matthe, H. Klessig, M. Simsek, G. Fettweis, J. Ansari, S. A. Ashraf, B. Almeroth, J. Voigt, I. Riedel *et al.*, "Latency critical IoT applications in 5G: perspective on the design of radio interface and network architecture," *IEEE Communications Magazine*, vol. 55, no. 2, pp. 70–78, Feb. 2017.

[2] T. Baker, M. Asim, H. Tawfik, B. Aldawsari, and R. Buyya, "An energy-aware service composition algorithm for multiple cloud-based iot applications," *Elsevier Journal of Network and Computer Applications*, vol. 89, pp. 96–108, July 2017.

[3] M. Díaz, C. Martín, and B. Rubio, "State-of-the-art, challenges, and open issues in the integration of internet of things and cloud computing," *Elsevier Journal of Network and Computer applications*, vol. 67, pp. 99–117, May 2016.

[4] "Cisco delivers vision of fog computing to accelerate value from billions of connected devices. press release. cisco." Jan. 2014, accessed on 04th Apr., 2017. [Online]. Available: https://newsroom.cisco.com/press-release-content?type=webcontent& articleId=1334100

[5] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, Mar. 2017.

[6] K. Dolui and S. K. Datta, "Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing," in *proc. IEEE Global Internet of Things Summit (GIoTS)*, Aug. 2017, pp. 1–6.

[7] C. Chang, S. N. Srirama, and R. Buyya, "Internet of things (IoT) and new computing paradigms," *Fog and Edge Computing: Principles and Paradigms*, pp. 1–23, Feb. 2019.

[8] M. Mukherjee, L. Shu, and D. Wang, "Survey of fog computing: Fundamental, network applications, and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 1826–1857, Mar. Mar. 2018.

[9] P. G. V. Naranjo, Z. Pooranian, M. Shojafar, M. Conti, and R. Buyya, "FOCAN: A fog-supported smart city network architecture for management of applications in the internet of everything environments," *Elsevier Journal of Parallel and Distributed Computing*, vol. 132, pp. 274–283, Oct. 2019.

[10] R. Mahmud, S. N. Srirama, K. Ramamohanarao, and R. Buyya, "Quality of experience (QoE)-aware placement of applications in fog computing environments," *Elsevier Journal of Parallel and Distributed Computing*, vol. 132, pp. 190–203, Oct. 2019.

[11] K. Kaur, N. Kumar, S. Garg, and J. J. Rodrigues, "EnLoc: Data locality-aware energy-efficient scheduling scheme for cloud data centers," in *Proc. IEEE International Conference on Communications (ICC)*, 2018, pp. 1–6.

[12] R. Buyya, S. N. Srirama, G. Casale *et al.*, "A manifesto for future generation cloud computing: research directions for the next decade," *ACM computing surveys (CSUR)*, vol. 51, no. 5, pp. 105–143, Jan. 2019.

[13] D. A. Chekired and L. Khoukhi, "Multi-tier fog architecture: A new delay-tolerant network for IoT data processing," in *Proc. IEEE International Conference on Communications (ICC)*, July 2018, pp. 1–6.

[14] S.-W. Ko, K. Huang, S.-L. Kim, and H. Chae, "Live prefetching for mobile computation offloading," *IEEE Transactions on Wireless Communications*, vol. 16, no. 5, pp. 3057–3071, Mar. 2017.

[15] Y. Yang, Z. Liu, X. Yang, K. Wang, X. Hong, and X. Ge, "POMT: Paired offloading of multiple tasks in heterogeneous fog networks," *IEEE Internet of Things Journal*, June 2019.

[16] Y. Wang, X. Tao, X. Zhang, P. Zhang, and Y. T. Hou, "Cooperative task offloading in three-tier mobile computing networks: An ADMM framework," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 3, pp. 2763–2776, Jan. 2019.

[17] M.-H. Chen, M. Dong, and B. Liang, "Resource sharing of a computing access point for multi-user mobile cloud offloading with delay constraints," *IEEE Transactions on Mobile Computing*, vol. 17, no. 12, pp. 2868–2881, Mar. 2018.

[18] S. Garg, A. Singh, K. Kaur, G. S. Aujla, S. Batra, N. Kumar, and M. S. Obaidat, "Edge computing-based security framework for big data analytics in VANETs," *IEEE Network*, vol. 33, no. 2, pp. 72–81, 2019.

[19] S. Garg, A. Singh, K. Kaur, S. Batra, N. Kumar, and M. S. Obaidat, "Edge-based content delivery for providing QoE in wireless networks using quotient filter," in *Proc. IEEE International Conference on Communications (ICC)*, 2018, pp. 1–6.

[20] G. Zhang, F. Shen, Z. Liu, Y. Yang, K. Wang, and M.-T. Zhou, "FEMTO: Fair and energy-minimized task offloading for fog-enabled IoT networks," *IEEE Internet of Things Journal*, Dec. 2018.

[21] M. Mukherjee, S. Kumar, M. Shojafar, Q. Zhang, and C. X. Mavromoustakis, "Joint task offloading and resource allocation for delay-sensitive fog networks," in *Proc. IEEE ICC*, May 2019, pp. 1–7.

[22] A. Yousefpour, G. Ishigaki, and J. P. Jue, "Fog computing: Towards minimizing delay in the internet of things," in *Proc. in international conference on edge computing (EDGE)*, Sept. 2017, pp. 17–24.

[23] V. B. C. Souza, W. Ramírez, X. Masip-Bruin, E. Marín-Tordera, G. Ren, and G. Tashakor, "Handling service allocation in combined fog-cloud scenarios," in *Proc. IEEE International Conference on Communications (ICC)*, May. 2016, pp. 1–5.

[24] T. G. Rodrigues, K. Suto, H. Nishiyama, and N. Kato, "Hybrid method for minimizing service delay in edge cloud computing through VM migration and transmission power control," *IEEE Transactions on Computers*, vol. 66, no. 5, pp. 810–819, Oct. 2016.

[25] C. Fricker, F. Guillemin, P. Robert, and G. Thompson, "Analysis of an offloading scheme for data centers in the framework of fog computing," *ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS)*, vol. 1, no. 4, p. 16, Sept. 2016.

[26] J. He, J. Wei, K. Chen, Z. Tang, Y. Zhou, and Y. Zhang, "Multitier fog computing with large-scale iot data analytics for smart cities," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 677–686, July 2017.

[27] Y.-Y. Shih, W.-H. Chung, A.-C. Pang, T.-C. Chiu, and H.-Y. Wei, "Enabling low-latency applications in fog-radio access networks," *IEEE network*, vol. 31, no. 1, pp. 52–58, Dec. 2016.

[28] 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, Oct. 2018.

[29] H. Shah-Mansouri and V. W. Wong, "Hierarchical fog-cloud computing for IoT systems: A computation offloading game," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 3246–3257, Aug. 2018.

[30] 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 *Proc. IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, Oct. 2018, pp. 154–159.

[31] J. Du, L. Zhao, J. Feng, and X. Chu, "Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee," *IEEE Transactions on Communications*, vol. 66, no. 4, pp. 1594–1608, Apr. 2018.

[32] Y.-L. Jiang, Y.-S. Chen, S.-W. Yang, and C.-H. Wu, "Energy-efficient task offloading for time-sensitive applications in fog computing," *IEEE Systems Journal*, Sept. 2019.

[33] 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 *Proc. 19th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, Nov. 2017, pp. 151–156.

[34] 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, Jan. 2019.

[35] Q. Fan and N. Ansari, "Workload allocation in hierarchical cloudlet networks," *IEEE Communications Letters*, vol. 22, no. 4, pp. 820–823, Feb. 2018.

[36] 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, June 2017.

[37] 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, Dec. 2017.

[38] 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, Oct. 2017.

[39] A. Alnoman and A. Anpalagan, "A dynamic priority service provision scheme for delay-sensitive applications in fog computing," in *Proc. 29th Biennial Symposium on Communications (BSC)*, Feb. 2018, pp. 1–5.

[40] U. Tadakamalla and D. Menascé, "FogQN: An analytic model for fog/cloud computing," in *Proc. IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)*, May 2018, pp. 307–313.

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. His area of research includes Internet of Things, fog Computing, Cloud Computing, Serverless Computing and Evolutionary algorithm. He has contributed numerous research Articles in various national and inter-national journal and conference.

Mithun Mukherjee received the Ph.D. degree in electrical engineering from the Indian Institute of Technology Patna, Patna, India, in 2015. Currently, an assistant professor with the Guangdong Provincial Key Laboratory of Petrochemical Equipment Fault Diagnosis, Guangdong University of Petrochemical Technology, Maoming, China. Dr. Mukherjee was a recipient of the 2016 EAI WICON, the 2017 IEEE SigTelCom, the 2018 IEEE Systems Journal, and the 2018 IEEE ANTS Best Paper Award. He has been a guest editor for IEEE Internet of Things Journal and IEEE Transactions on Industrial Informatics. His research interests include wireless communications, fog computing, and ultra-reliable low-latency communications.

Satish Narayana Srirama is a Research Professor and the head of the Mobile & Cloud Lab at the Institute of Computer Science, University of Tartu, Estonia and a Visiting Professor at University of Hyderabad, India. He received his PhD in computer science from RWTH Aachen University, Germany. His current research focuses on cloud computing, mobile web services, mobile cloud, Internet of Things, fog computing, migrating scientific computing and enterprise applications to the cloud and large-scale data analytics on the cloud. He is IEEE Senior Member, an Editor of Wiley Software: Practice and Experience, a 49 year old Journal, was an Associate Editor of IEEE Transactions in Cloud Computing and a program committee member of several international conferences and workshops. Dr. Srirama has co-authored over 130 refereed scientific publications in international conferences and journals.