LETO: An Efficient Load Balanced Strategy for Task Offloading in IoT-Fog Systems

Chittaranjan Swain, Manmath Narayan Sahoo, Anurag Satpathy
Department of Computer Science and Engineering
National Institute of Technology, Rourkela, India.
{chittaranjanswain518, anurag.satpathy}@gmail.com, sahoom@nitrkl.ac.in

***Abstract*—The resource-constrained IoT devices often offload tasks to Fog nodes (FNs) owing to the intermittent WAN delays and multi-hopping by executing at remote cloud servers. An efficient allocation strategy satisfies the users' requirements by ensuring minimum offloading delays and provides a balanced assignment from the service providers' (SPs) viewpoint. This paper presents a model called LETO that reduces the total offloading delay for real-time tasks and achieves a balanced assignment across FNs. The overall problem is modeled as a one-to-many matching game with maximum and minimum quotas. Owing to the deferred acceptance algorithm (DAA) inapplicability, we use a proficient version of the DAA called multi-stage deferred acceptance algorithm (MSDA) to obtain a *fair* and *Pareto-optimal* assignment of tasks to FNs. Extensive simulations confirm that LETO can achieve a more balanced assignment compared to the baseline algorithms.**

***Index Terms*—Load Balancing, Task Offloading, IoT, Fog Systems, Matching Theory, Max-Min Quota**

I. Introduction

Task offloading refers to delegating the execution of a service from a resource-constrained Internet of Things (IoT) device to a nearby Fog device or a remote cloud server. Offloading services to a remote cloud server often leads to a higher response time owing to intermittent WAN delays, multi-hopping, and scarce spectrum resources. However, offloading to Fog nodes (FNs) not only improves the users' response time but also provides additional benefits such as location awareness and real-time mobility support [1]. From a SP's viewpoint allocating resources to offloaded services is a complex operation owing to its heterogeneous demands and limited resources of FNs [2]. Moreover, for the increasing number of offloading requests, it is particularly challenging to simultaneously realize the desired quality of service (QoS) for real-time applications and balance the load of the FNs. Achieving both will improve the resource utilization of FNs and assist applications that include augmented reality (AR) and online gaming to realize tolerable latency.

The literature on task offloading focused independently on addressing QoS requirements of applications such as completion time and deadlines. The authors in [3] [4] [5] [6] modelled the offloading as optimization problem to reduce the latency of IoT services. Although optimization solvers may guarantee sub-optimal solutions, they suffer from the following pitfalls. Firstly, they focus on system-wide objectives that may not align with the objectives of individual stakeholders. Secondly, they are computationally expensive and non-scalable. Matching theory-based solutions overcome these drawbacks and are primarily focused on reducing the completion time, energy consumption, and outages, i.e., the number of tasks overshooting their deadlines [1] [7]. However, Zhang *et al.* [8] pointed out that an unbalanced assignment may create a bottleneck of computational resources at certain FNs, thereby causing QoS violations. Hence, it is essential to incorporate load balancing mechanisms while generating an offloading schedule without compromising with the offloading delay of services. These can be achieved by enforcing minimum quotas at the FNs. Although the deferred acceptance algorithm (DAA) generates efficient assignments, it is incapable of achieving an assignment satisfying the minimum quota of FNs [9]. Hence, in this paper, we utilize a variation of DAA called multi-stage deferred acceptance algorithm (MSDA) to achieve the aforementioned objectives. The overall contributions of the paper are as follows:

- We propose a model called *LETO* that aims to reduce the total completion/offloading delay and outages from the users' perspective. In the viewpoint of a SP, *LETO* aims at achieving a balanced assignment across FNs.
- The offloading game is modeled as a one-to-many matching game with minimum and maximum quota at the FNs.
- To validate the performance of *LETO*, we perform extensive simulations and compare its effectiveness with two different baselines: Highest Data Rate (HDR) and Highest Computing Device (HCD) [10]. Simulation results confirm that *LETO* can achieve a more balanced assignment across baselines for all test cases.

The rest of the paper is organized as follows. Section II discusses the literature that we have reviewed. In Section III and IV, we discuss the system model and solution approach in detail. Performance analysis of *LETO* is discussed in Section V and conclusions are drawn in Section VI.

II. Related Work

In this section, we discuss the literature devoted to full offloading in IoT-Fog interconnection networks. Primarily the works independently address latency and deadline concerns of hosted IoT services. Considering latency, Chitti *et al.* [11] discussed a matching theory-based framework to minimize the worst-case service latency incurred in executing offloaded tasks at the FNs. Alternatively, Yousefour *et al.* [12] presented an analytical model to reduce the service latency of offloaded

978-1-6654-1681-8/21/$31.00 ©2021 IEEE
DOI 10.1109/ICWS53863.2021.00065

services in a densely connected IoT-Fog-Cloud environment. Some other works that have focused on reducing the latency in different offloading environments are discussed in [3] [4]. All the above strategies focus on independently reducing latency that may not adhere to the stringent deadline requirements of real-time applications. As a remedy, some offloading strategies to concurrently minimize latency as well as outages are discussed in [1] [7] [10].

With the rapid growth of IoT services and varying degrees of requirements, offloading techniques may often face unbalanced assignments. This can have a deleterious impact on *user satisfaction*, *availability of resources*, and *utilization of FNs*. Hussein and Mousa [13] proposed an ant colony optimization (ACO) and particle swarm optimization (PSO) based hybrid technique to simultaneously achieve a balanced assignment and reduced service latency in mobile edge environments. Although the technique can achieve sub-optimality, it suffers from lacuna of the optimization techniques as discussed in Section I. It can be observed from the reviewed literature that not a lot of research has gone into developing efficient strategies to obtain a balanced assignment without compromising the latency and deadline requirements of user services. Therefore, in this work, we propose a one-to-many matching framework based on a multi-stage deferred acceptance algorithm (MSDA) to achieve all the above-mentioned objectives. Next, we provide a detailed discussion on the system model followed by the solution strategy.

III. System Model and Assumptions

The overall architecture of an interconnected Fog network is depicted in Fig. 1. It consists of 'm' IoT devices where each device d_i generates a task t_i and the set of all tasks is captured by $\mathcal{T} = \{t_1, t_2, t_3, \cdots, t_m\}$. We assume that the device d_i is resource-constrained and is incapable of executing t_i locally. Hence, t_i is to be offloaded to one of the FNs in $\mathcal{F} = \{f_1, f_2, f_3, \cdots f_n\}$ for successful execution [1]. The offloading assignment is taken care of by the *service broker* (SB). The offloading request of a task t_i is captured as a

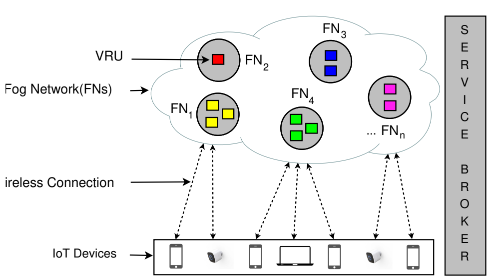

Fig. 1. IoT-Fog System Architecture

quadruple $< s_i, c_i, d_i, \tau_i >$. Here, s_i and τ_i denote the input and output size (*bits*), and c_i and d_i correspond to the computational demand (*cycles*) and deadline (*s*). The computational resources at a FN are logically partitioned into independent executable entities called virtual resource units ($VRUs$) [1]. Let γ_j be the computation capabilities of the homogeneous $VRUs$ at FN f_j, expressed in *cycles/sec*. However, $VRUs$ at different FNs are heterogeneous, i.e., $\gamma_j \neq \gamma_{j'}$, $j \neq j'$ and $j, j' \in [1, n]$. It is considered that a *VRU* can execute one task at a time. The number of *VRUs* at f_j, denoted as q_j, is called as its *maximum quota*. It reflects the maximum number of tasks that can be executed in parallel. To obtain a balanced assignment of tasks across the FNs a *minimum quota* p_j is imposed at each FN $f_j \in \mathcal{F}$. It indicates the minimum number of $VRUs$ of a FN that should be utilized in any balanced assignment.

As discussed previously, IoT devices are resource-constrained and are dependent on nearby FNs for the real-time execution of tasks. The offloading procedure executes in two phases, viz., (*i.*) *communication phase* and (*ii.*) *execution phase*. The *communication phase* involves transmitting a task to a FN and retrieving the processed results following successful execution. The time consumed in this phase is termed as *communication delay*. The *execution phase* focuses on the successful execution of an offloaded task at a FN. The time incurred in this phase is called as *execution delay*. Therefore, the *offloading delay* of a task is an aggregate latency incurred in the aforementioned phases.

A. Computation of Communication Delay

The communication delay comprises (*i.*) transmission delay and (*ii.*) receiving delay. Transmissions are carried out over a noisy wireless channel [1].

Transmission delay: It is the time required to transfer a task t_i from a device d_i to a designated FN f_j for computation. It is considered that each device d_i has an active uplink and downlink channel to a FN f_j with bandwidth $B_{i,j}$ and $B_{j,i}$ respectively. Let p_i^t be the transmission power of d_i, $h_{i,j}$ represents the channel gain between d_i and f_j, and n_o is the noise power. Therefore, the maximum achievable uplink data rate $R_{i,j}$ between d_i and f_j can be computed as per Eq. (1).

$$R_{i,j} = B_{i,j} * \log_2(1 + \frac{p_i^t * h_{i,j}}{n_0}) \quad (1)$$

Considering the uplink data rate $R_{i,j}$ and input size s_i of task t_i, the transmission delay $D_{i,j}^{trans}$ from d_i to f_j is calculated as per Eq. (2).

$$D_{i,j}^{trans} = \frac{s_i}{R_{i,j}} \quad (2)$$

Receiving delay: It is the amount of time required to receive the processed output at d_i, after successful execution of t_i at f_j. Given the transmission power p_j^t of f_j, and the channel gain $h_{j,i}$ between f_j and d_i, the maximum achievable downlink data rate $R_{j,i}$ between f_j and d_i can be calculated as per Eq. (3).

$$R_{j,i} = B_{j,i} * \log_2(1 + \frac{p_j^t * h_{j,i}}{n_0}) \quad (3)$$

Taking into account the downlink data rate $R_{j,i}$ and output size τ_i of t_i, the receiving delay $D_{j,i}^{rcv}$ from f_j can be calculated using Eq. (4).

$$D_{j,i}^{rcv} = \frac{\tau_i}{R_{j,i}} \quad (4)$$

B. Computation of Execution Delay

The delay $D_{i,j}^{exe}$ in executing an offloaded task t_i at f_j is dependent on the computational demand c_i of t_i and computational capability γ_j of a VRU at f_j. It can be calculated as per Eq. (5).

$$D_{i,j}^{exe} = \frac{c_i}{\gamma_j} \tag{5}$$

C. Offloading Delay

The offloading delay also termed as completion delay, denoted by $D_{i,j}^{off}$, is the aggregate of transmission, execution, and receiving delays and can be expressed as Eq. (6).

$$D_{i,j}^{off} = D_{i,j}^{trans} + D_{i,j}^{exe} + D_{j,i}^{rcv} \tag{6}$$

D. Problem Formulation

Let $x_{i,j}$ be a binary indicator variable that signifies if a task t_i is assigned to f_j or not. This is captured as Eq. (7).

$$x_{i,j} = \begin{cases} 1 : \text{if } t_i \text{ is matched to } f_j \\ 0 : \text{otherwise} \end{cases} \tag{7}$$

We also define $\mathcal{O}$ to be the set of tasks suffering from outage and can be computed as per Eq. (8).

$$\mathcal{O} = \{t_i \in \mathcal{T} \mid x_{i,j} = 1 \ \& \ D_{i,j}^{off} > d_i; j \in [1, n]\} \tag{8}$$

The overall objective of *LETO* is to minimize the total offloading delay and the number of outages and is expressed in Eq. (9a). A task can be assigned to only one FN and is reflected in Constraint 9b. Constraint 9c ensures that a FN $f_j \in \mathcal{F}$ should be assigned at least p_j and at most q_j tasks in any mapping. Enforcing this enables the SP to distribute the load across the FNs in the Fog network. If the total number of tasks $|\mathcal{T}| > \sum_{j=1}^{n} q_j$ or $|\mathcal{T}| < \sum_{j=1}^{n} p_j$, then there is no possible way to perform an assignment without violating quota [14]. Therefore, to obtain a feasible assignment, Constraint 9d should never be violated. Finally, Constraint 9e indicates the acceptable range of values the variables can take.

$$min \left(\sum_{i=1}^{m} \sum_{j=1}^{n} x_{i,j} * D_{i,j}^{off} \right) \ and \ (|\mathcal{O}|) \tag{9a}$$

$$s.t. \quad \sum_{j=1}^{n} x_{i,j} = 1 \tag{9b}$$

$$p_j \le \sum_{i=1}^{m} x_{i,j} \le q_j; \quad p_j, q_j > 0 \tag{9c}$$

$$\sum_{j=1}^{n} p_j \le |\mathcal{T}| \le \sum_{j=1}^{n} q_j \tag{9d}$$

$$\forall\, i \in [1, m], \quad \forall\, j \in [1, n] \tag{9e}$$

The overall problem expressed in Eq. (9a) is proven to be $\mathcal{NP}$-Hard [1]. Therefore, we propose a one-to-many matching-based strategy called *LETO* to achieve a Pareto-optimal assignment in polynomial-time and is discussed subsequently.

IV. Solution Approach

Matching theory is an elegant and efficient technique to perform assignments between distinct sets of agents by considering their individual preferences [15]. Preferences reflect the level of satisfaction of each agent in the matching and are autonomously generated.

A. Task Offloading as a Matching Game

Formally, the offloading game can be expressed through the following Definitions

Definition 1. *Let $\mathcal{T}$ and $\mathcal{F}$ be the set of tasks and FNs. A matching game defined over ($\mathcal{T}$, $\mathcal{F}$) has two preference relations $\succ_{t_i}$ and $\succ_{f_j}$ that allows each agent $t_i \in \mathcal{T}$ to specify preference over all agents $f_j \in \mathcal{F}$, and vice-versa.*

Definition 2. *The offloading game is defined by a one-to-many matching function $\mu : \mathcal{T} \cup \mathcal{F} \rightarrow 2^{\mathcal{T} \,\cup\, \mathcal{F}}$ such that*

$$\mu(t_i) \subseteq \mathcal{F} \ and \ |\mu(t_i)| = 1 \tag{10a}$$

$$\mu(f_j) \subseteq \mathcal{T} \ and \ |\mu(f_j)| \le q_j \tag{10b}$$

$$f_j \in \mu(t_i) \Leftrightarrow t_i \in \mu(f_j) \tag{10c}$$

Condition (10a) states that each task is assigned to exactly one FN. A FN f_j can host a maximum of q_j tasks which is reflected by Condition (10b). A task t_i is mapped to a FN f_j *iff* f_j is assigned task t_i. This is expressed in Condition (10c).

Definition 3. *The matching μ is said to be blocked by a task and FN pair (t_i, f_j) if it satisfies the following.*

$$t_i \notin \mu(f_j) \tag{11a}$$

$$f_j \succ_{t_i} \mu(t_i) \ and \ t_i \succ_{f_j} t_{i'},\, t_{i'} \in \mu\,(f_j) \tag{11b}$$

Condition (11a) reflects that the task t_i is not mapped to f_j. Condition (11b) states that t_i prefers f_j over its current assignment $\mu(t_i)$ and f_j prefers t_i over $t_{i'}$ such that $t_{i'} \in \mu\,(f_j)$. In such case t_i and f_j have incentive to deviate from their current assignments and form a blocking pair.

Definition 4. *The matching μ is said to be stable if there exists no blocking pair.*

The deferred acceptance algorithm (DAA) has been successfully used to obtain a *stable* assignment in different scenarios [1] [11]. However, the DAA fails to achieve a *stable* assignment when imposed with minimum quota [14]. Hence, we solve the load balanced offloading game using a modified version of DAA called multi-stage deferred acceptance algorithm (MSDA).

B. LETO: A Load-Balanced Task Offloading Strategy

LETO works in two phases, i.e., (*i.*) preference generation and (*ii.*) stable assignment using MSDA. Before discussing the details of each phase, we define important terms such as *feasibility*, *Pareto-optimality*, and *fairness*.

Definition 5. *The matching μ is said to be feasible iff it satisfies all the conditions of Definition 2 and the following additional condition.*

$$p_j \leq \ |\mu(f_j)| \ \leq q_j \tag{12}$$

Condition (12) enforces feasibility by ensuring that each FN f_j is mapped to at least p_j and at most q_j tasks in any matching.

Definition 6. *A matching μ is said to be Pareto-optimal if there does not exist another feasible matching μ' such that $\mu'(t_i) \succeq_{t_i} \mu(t_i)$, $\forall\ t_i \in \mathcal{T}$ and $\exists\ t_i \in \mathcal{T}$ satisfying $\mu'(t_i) \succ_{t_i} \mu(t_i)$.*

Definition 7. *A matching μ is said to be PL-blocked by a pair of agents (t_i, f_j) iff it satisfies the conditions of Definition 3 and the following condition*

$$t_i \succ_{PL} t_{i'};\ i \neq i' \tag{13}$$

The standard definition of blocking pair in Definition 3 produces too many blocking pairs when DAA is imposed with minimum quota [9] [14]. Since establishing a fair assignment is of primary importance, some of these potential blocking pairs must be invalidated. Therefore, a new notion of PL-fairness invalidating such pairs is introduced [14].

Definition 8. *The matching function μ is said to be PL-fair iff it is feasible and is not PL-blocked by any pair of agents.*

1) Preference Generation: The preference profiles of all agents are *complete*, *strict*, and *transitive*.
Preference Profiles of Tasks: Each task t_i sets preference list $P(t_i)$ that ranks all $f_j \in \mathcal{F}$ considering the offloading delay computed as per Eq. (6). Therefore,

$$f_j \succ_{t_i} f_{j'} \iff D_{i,j}^{off} < D_{i,j'}^{off};\ j \neq j'$$

Preference Profiles of FNs: Each FN f_j prepares a preference profile $P(f_j)$ for all tasks in $\mathcal{T}$ based on their deadlines. Therefore,

$$t_i \succ_{f_j} t_{i'} \iff d_i < d_{i'};\ i \neq i'$$

Precedence List: The precedence list provides a ranking of all the tasks in $\mathcal{T}$. To achieve *pareto optimality* the ordering of tasks in the PL follows the preference profile of the FNs, i.e., tasks in the PL are sorted as per their deadlines [9]. Therefore,

$$t_i \succ_{PL} t_{i'} \iff d_i < d_{i'};\ i \neq i'$$

2) Working of MSDA: The overall working of MSDA is shown in Algorithm 1. The algorithm takes as input the preferences of each agent in $\mathcal{T}$ and $\mathcal{F}$, maximum and minimum quotas of each $f_j \in \mathcal{F}$ denoted by q_j and p_j, and a PL containing the ordering of all the tasks. The algorithm outputs a *pareto-optimal* assignment μ that maps each task $t_i \in \mathcal{T}$ to exactly one FN and each FN $f_j \in \mathcal{F}$ to at least p_j tasks. Initially, all the tasks are free, i.e., $free[i] = True$ and $\mathbb{R}^0$ is initialized to PL. The maximum and minimum quota of f_j in k^{th} stage of MSDA, respectively denoted as q_j^k and p_j^k, are initialized to q_j and p_j. The algorithm then reserves $\mathbb{R}^k$ as the set of least preferred r^k tasks from PL (Steps 3-4). After reservation, depending on the remaining tasks in $\mathbb{R}^{k-1} \setminus \mathbb{R}^k$ two cases may arise: (*i.*) if remaining task set is non-empty then CDAA (Algorithm 2) is called on $\mathbb{R}^{k-1} \setminus \mathbb{R}^k$ with maximum quotas, (*ii.*) if it is empty, CDAA is invoked on $\mathbb{R}^k$ with minimum quotas (Steps 5-11). The former case allows the unreserved tasks to propose and possibly assigned to their preferred FNs. The latter case ensures that each FN satisfies its minimum quota constraint. Algorithm 2 provides an assignment to the proposing tasks at stage k of MSDA. It enters into the proposal phase that allows all free tasks $t_i \in \mathbb{T}$, in order of PL, with a non-empty preference list $P(t_i)$ to send

Algorithm 1: Multistage Deferred Acceptance Algorithm (MSDA)

Input: PL, $P(t_i)$, $P(f_j)$, p_j, q_j, $\forall t_i \in \mathcal{T}$, $\forall f_j \in \mathcal{F}$
Result: $\mu : \mathcal{T} \cup \mathcal{F} \rightarrow 2^{\mathcal{T} \cup \mathcal{F}}$
1 **Initialize**: $free[i] = True$, $\forall i \in [1, m]$, $\mu = \phi$
    $\mathbb{R}^0 = PL$, $k = 1$, $p_j^k = p_j$, $q_j^k = q_j$, $\forall j \in [1, n]$
2 **while** $PL \neq \Phi$ **do**
3     $r^k = \sum_{j=1}^{n} p_j^k$
4     $\mathbb{R}^k = \{t_{m-r^k+1}, t_{m-r^k+2}, ..., t_m\}$, Here $\mathbb{R}^k$ is the set of r^k tasks with lower preference according to PL
5   **if** $\mathbb{R}^{k-1} \setminus \mathbb{R}^k \neq \phi$ **then**
6         $\mathbb{T} = \mathbb{R}^{k-1} \setminus \mathbb{R}^k$
7         $s_j^k = q_j^k$, $j \in [1, n]$
8   **else**
9         $\mathbb{T} = \mathbb{R}^k$
10         $s_j^k = p_j^k$, $\forall j \in [1, n]$
11   μ^k = CDAA ($\mathbb{T}$, $\mathcal{F}$, $\{s_j^k\}_{j \in [1,n]}$)
12     $\mu = \mu \cup \mu^k$
13   **for** *each* $f_j \in \mathcal{F}$ **do**
14         $q_j^{k+1} = q_j^k - |\mu^k(f_j)|$
15         $p_j^{k+1} = max\{0, p_j^k - |\mu^k(f_j)|\}$
16         $PL = PL \setminus \mu^k(f_j)$
17     $k = k + 1$

Algorithm 2: Classical Deferred Acceptance Algorithm (CDAA)

Input: $P(t_i)$, $\forall t_i \in \mathbb{T}$, s_j^k, $P(f_j)$, $\forall j \in [1, n]$
Result: μ^k
1 **Initialize** $\mu^k = \phi$
2 **while** $\exists\, t_i$ *|free[i] and* $P(t_i) \neq \phi$ **do**
3     $f_{j'}$ = highest ranked FN in $P(t_i)$ to which t_i has not yet proposed.
4   Send proposal to $f_{j'}$
5   **if** $s_{j'}^k > 0$ **then**
6         $\mu^k = \mu^k \cup \{(t_i, f_{j'})\}$
7         $free[i] = False$
8   **else**
9   Reject t_i
10 return μ^k

out proposals to their most preferred FN $f_{j'}$ (Steps 3-4). On receiving a proposal form t_i, two scenarios may arise at FN $f_{j'}$: (*i.*) it has sufficient quota to match t_i and $f_{j'}$ or (*ii.*) it does not have sufficient quota and t_i is rejected (Steps 5-9). Once CDAA returns the matching μ^k to MSDA, the minimum and maximum quotas are updated. The newly matched tasks are added to μ and removed from the PL (Steps 12-16, Algorithm 1). The algorithm terminates when all tasks are assigned, i.e., $PL = \phi$.

V. Performance Evaluation

We have performed simulations using the iFogSim simulator [16]. The environmental setup and analysis of the simulation results are discussed elaborately in this Section.

A. Environmental Setup

The FNs and IoT devices are statically deployed. The distance between IoT device and FN is uniformly distributed as U$[50, 500]$ m and channel bandwidth between them is 20 *MHz* [7]. There are 4 FNs, and the computational capability of each is expressed in the form of VRUs which are generated following a uniform distribution U$[150, 350]$ that represents its maximum quota. The minimum quota at each FN is set randomly in the interval $[0, \left\lfloor \frac{|\mathcal{T}|}{|\mathcal{F}|} \right\rfloor]$ [9]. The computational rate (*cycles/s*) is chosen in the range U$[6, 10]$ *GHz* [1]. The number of IoT devices varies in the range of $250 - 1000$ at an interval of 250 per observation. The task specific parameters such as input size, output size, computational demand, and deadline are generated uniformly in the range U$[300, 600]$ *Kb*, U$[10, 20]$ *Kb*, U$[210, 480]$ *million cycles*, and U$[7, 25]$ *s*, respectively [1]. Considering PCS-1900 GSM band, the free space path loss in *dB* between an IoT device d_i and a FN f_j is calculated as $pl_{d_i,f_j} = 38.02 + 20log(dist(d_i, f_j))$, where $dist(d_i, f_j)$ is the distance between d_i and f_j [1]. The channel gain is then calculated as $h_{i,j} = 10^{-(pl_{d_i,f_j})/10}$. The transmission power of an IoT device and noise power of channel is set to 0.5 *W* and 10^{-10} *W* respectively [1] [7].

B. The Baseline Algorithms

To compare the performance of *LETO*, we assess its performance against the following baseline algorithms: (*i.*) *Highest Data Rate (HDR)* [10], where IoT device greedily offloads the tasks to a FNs with minimum transmission delay and (*ii.*) *Highest Computing Device (HCD)* [10] where tasks are assigned to available FNs with highest computational speed.

C. Simulation Results

Fig. 2 depicts the total offloading delay for executing $[250, 500, 750, 1000]$ tasks. As expected the total offloading delay increases with an increasing number of tasks. For 250 tasks *LETO* has a slightly higher offloading delay compared to HCD. This is because HCD greedily allocates tasks to the available FNs with the best computing capabilities, whereas *LETO* focuses on a balanced assignment across FNs by satisfying their minimum quota, thereby resulting in some

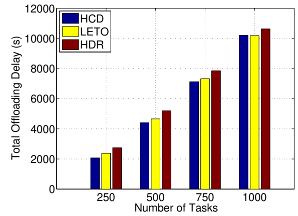

Fig. 2. Offloading Delay (s) Vs. Number of Tasks.

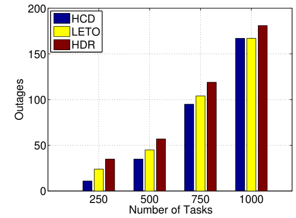

Fig. 3. Outages Vs. Number of Tasks.

tasks getting assigned to computationally less efficient FNs. For a larger test case a small quanta of tasks are forced to choose some less preferred FNs, with increasing number of tasks in MSDA. In other words, a majority of tasks are assigned to the best possible FNs. Hence, *LETO* has a slightly higher offloading delay compared to HCD. The HDR strategy performs poorly compared to HCD and *LETO* as it greedily selects the nearest FN without considering the computing capabilities of the selected FN.

Fig. 3 displays the total number of outages for different strategies considering different evaluation scenarios. The number of outages increases with an increasing number of tasks as the individual offloading delay per task increases. Once the maximum quota of the better performing FNs are exhausted, the unassigned tasks are forced onto computationally less efficient FNs, thereby elevating their offloading delays. The comparative behavior of HCD and *LETO* concerning outages is similar to the offloading delay comparison discussed previously. HDR, on the other hand, does not consider deadline as a parameter for assignment thereby leading to a higher number of outages.

Fig. 4 shows the utilization of different FNs for executing 250 tasks. It can be observed from the figure that due to the enforcement of minimum quota at each FN, *LETO* is able to achieve a more balanced assignment compared to HCD and HDR. HCD faces the issue of clustering at $f4$ as it is the most efficient FN considering its computing capabilities. On the other hand, in HDR the assignment is dependent on the distance to the FNs. Hence, we observe a distributed assignment across FNs. Therefore, HDR suffers from a higher offloading delay and consequently more outages compared to *LETO*. Figs 5 and 6 highlight the utilization of different FNs for 500 and 750 tasks respectively. The resource utilization of FNs increase with increasing number of tasks for all strategies. In contrast to Fig. 4, owing to higher number of tasks and limited $VRUs$ at $f4$, the additional tasks are allotted to the next computationally best FNs, i.e., $f3$ followed by $f2$, in HCD. Similar to Fig. 4 the allocation in HDR is distributed. Finally, the resource utilization for 1000 tasks is depicted in Fig. 7. It can observed that *LETO* and HCD have similar utilization levels as maximum quota of best performing FNs in the order, i.e., $f4$, $f3$, and $f2$ are filled up whereas the quota of the worst FN $f1$ only meets its minimum quota requirement. The maximum number of tasks in the experiment is equal to

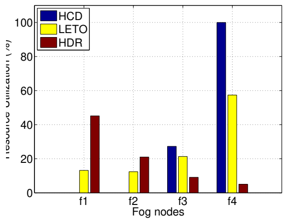

Fig. 4. Resource Utilization (%) Vs. FNs (For 250 tasks).

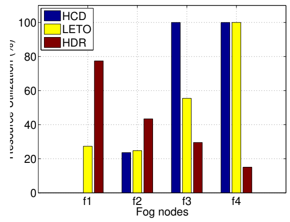

Fig. 5. Resource Utilization (%) Vs. FNs (For 500 tasks)

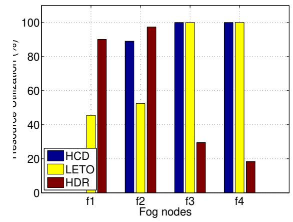

Fig. 6. Resource Utilization (%) Vs. FNs (For 750 tasks)

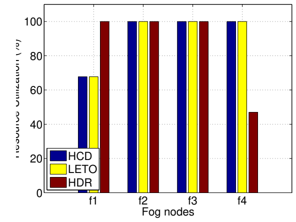

Fig. 7. Resource Utilization (%) Vs. FNs (For 1000 tasks)

the total number of $VRUs$ across the FNs, in which case all the $VRUs$ are utilized irrespective of any strategy. This can be observed from Figure 7.

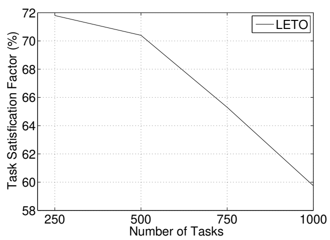

Fig. 8. Task Satisfaction Factor (s) Vs. Number of Tasks.

Fig. 8 illustrates the average satisfaction factor of each task in the matching. It can be observed that the satisfaction level of each task decreases with increasing number of tasks. The reason for this is two fold: (*i.*) tasks are forced onto less preferred FNs for meeting minimum quota, (*ii.*) once the best performing FNs are allotted their designated maximum quota, the tasks that appear later in the PL are mapped to their less preferred FNs. With increase in number of tasks and limited $VRUs$ at best FNs, high percentage of tasks are pushed to be mapped to less preferred FNs which leads to unsatisfactory assignments. This leads to a sharp decline in task satisfaction factor for larger test cases.
Thus the obtained simulation results confirm that *LETO* can achieve a more balanced assignment compared to the baseline algorithms.

VI. Conclusion

In this paper, we proposed a model called *LETO* to achieve an efficient and balanced assignment of tasks to FNs in a densely connected IoT-Fog network. The offloading problem is formulated as a one-to-many matching game with minimum and maximum quotas and solve it using a multi-stage deferred acceptance algorithm (MSDA). To validate the efficiency of *LETO*, we compare its performance with two different baseline algorithms. Thorough simulation analysis confirms that *LETO* is able to achieve a more balanced assignment across baselines considering all test cases. As a consequence of a balanced assignment, *LETO* suffers from a marginal increase in offloading delay and number of outages compared to HCD for smaller test cases.

References

[1] C. Swain, M. N. Sahoo, A. Satpathy, K. Muhammad, S. Bakshi, J. J. P. C. Rodrigues, and V. H. C. de Albuquerque, "Meto: Matching theory based efficient task offloading in iot-fog interconnection networks," *IEEE Internet of Things Journal*, pp. 1–1, 2020.

[2] Y. Gu, Z. Chang, M. Pan, L. Song, and Z. Han, "Joint radio and computational resource allocation in iot fog computing," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 8, pp. 7475–7484, 2018.

[3] S. A. Zakaryia, S. A. Ahmed, and M. K. Hussein, "Evolutionary offloading in an edge environment," *Egyptian Informatics Journal*, 2020.

[4] P. Sun, H. Zhang, H. Ji, and L. Xi, "Small cells clustering and resource allocation in dense network with mobile edge computing," in *2019 IEEE Wireless Communications and Networking Conference (WCNC)*, 2019, pp. 1–6.

[5] P. L. Nguyen, R. H. Hwang, P. M. Khiem, K. Nguyen, and Y. D. Lin, "Modeling and minimizing latency in three-tier v2x networks," in *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, 2020, pp. 1–6.

[6] 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 *2016 IEEE international conference on communications (ICC)*. IEEE, 2016, pp. 1–5.

[7] C. Swain, M. N. Sahoo, and A. Satpathy, "Spato: A student project allocation based task offloading in iot-fog systems," *arXiv preprint arXiv:2105.10715*, 2021.

[8] J. Zhang, H. Guo, J. Liu, and Y. Zhang, "Task offloading in vehicular edge computing networks: A load-balancing solution," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 2, pp. 2092–2104, 2020.

[9] O. Semiari, W. Saad, and M. Bennis, "Downlink cell association and load balancing for joint millimeter wave-microwave cellular networks," in *2016 IEEE Global Communications Conference (GLOBECOM)*, 2016, pp. 1–6.

[10] M. Adhikari, M. Mukherjee, and S. N. Srirama, "Dpto: A deadline and priority-aware task offloading in fog computing framework leveraging multilevel feedback queueing," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 5773–5782, 2020.

[11] F. Chiti, R. Fantacci, and B. Picano, "A matching theory framework for tasks offloading in fog computing for iot systems," *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 5089–5096, 2018.

[12] A. Yousefpour, G. Ishigaki, R. Gour, and J. P. Jue, "On reducing iot service delay via fog offloading," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 998–1010, 2018.

[13] M. K. Hussein and M. H. Mousa, "Efficient task offloading for iot-based applications in fog computing using ant colony optimization," *IEEE Access*, vol. 8, pp. 37 191–37 201, 2020.

[14] D. Fragiadakis, A. Iwasaki, P. Troyan, S. Ueda, and M. Yokoo, "Strategyproof matching with minimum quotas," vol. 4, no. 1, 2016. [Online]. Available: https://doi.org/10.1145/2841226

[15] A. Satpathy, M. N. Sahoo, L. Behera, C. Swain, and A. Mishra, "Vmatch: A matching theory based vdc reconfiguration strategy," in *2020 IEEE 13th International Conference on Cloud Computing (CLOUD)*, 2020, pp. 133–140.

[16] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya, "ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments," *Software: Practice and Experience*, vol. 47, no. 9, pp. 1275–1296, 2017.