Multi-Armed Bandit-based Decentralized Computation Offloading in Fog-Enabled IoT

Sudip Misra, *Senior Member, IEEE*, Sri Pramodh Rachuri, *Student Member, IEEE*, Pallav Kumar Deb, *Graduate Student Member, IEEE*, and Anandarup Mukherjee, *Graduate Student Member, IEEE*

Abstract—The Internet of Things (IoT) environments have hard real-time tasks that need execution within fixed deadlines. As IoT devices consist of a myriad of sensors, each task is composed of multiple interdependent subtasks. Towards this, the cloud and fog computing platforms have the potential of facilitating these IoT Sensor Nodes (SNs) in accommodating complex operations with minimum delay. To further reduce operational latencies, we breakdown the high-level tasks into smaller subtasks and form a Directed Acyclic Task Graph (DATG). Initially, the SNs offload their tasks to a nearby Fog Node (FN) based on a greedy choice. The greedy formulation helps in selecting the FN in linear time while avoiding combinatorial optimizations at the SN, which saves time as well as energy. IoT environments are highly dynamic, which mandates the need for adaptive solutions. At the chosen FN, depending on the dependencies on the DATGs, its corresponding deadlines, and the varying conditions of the other FNs, we propose an ϵ-greedy non-stationary multiarmed bandit-based scheme (D2CIT) for online task allocation among them. The online learning D2CIT scheme allows the FN to autonomously select a set of FNs for distributing the subtasks among themselves and executes the subtasks in parallel with minimum latency, energy, and resource usage. Simulation results show that D2CIT offers a reduction in latency by 17% compared to traditional fog computing schemes. Additionally, upon comparison with existing online learning-based task offloading solutions in fog environments, D2CIT offers an improved speedup of 59% due to the induced parallelism.

Index Terms—Computation Offloading, Fog Computing, Distributed and Parallel Computing, Internet of Things, Reinforcement Learning.

1 Introduction

The advancement of the Internet of Things (IoT) and its applications has led to the generation of sizeable tasks. The user devices/Sensor Nodes (SNs) in IoT scenarios have a wide range of sensors on board, such as cameras, GPS, Infrared, and others. These sensors generate sizeable data, which requires complex operations for inferencing. Operations on these data involve different types of tasks ranging from simple actuation to intensive image processing. Further, these hard real-time applications need results within deadlines [1]. Some mission-critical applications also need computation in the order of milliseconds. The increasing number of SNs leads to an increase in the demand for operational services by external platforms like the Cloud and Fog. Existing literature concerning fog computing has successfully brought services like IaaS, PaaS, and SaaS closer to the edge of the network, which reduces latency, network bandwidth, as well as energy [2]. In this work, we further decrease the operational latency by assuming that the high-level tasks of the SNs are in the form of Directed Acyclic Task Graphs (DATGs), and performing executions on the subtasks in parallel. Currently, we focus only on the selection of the set of FNs for distribution and parallel processing to complete tasks within deadlines.

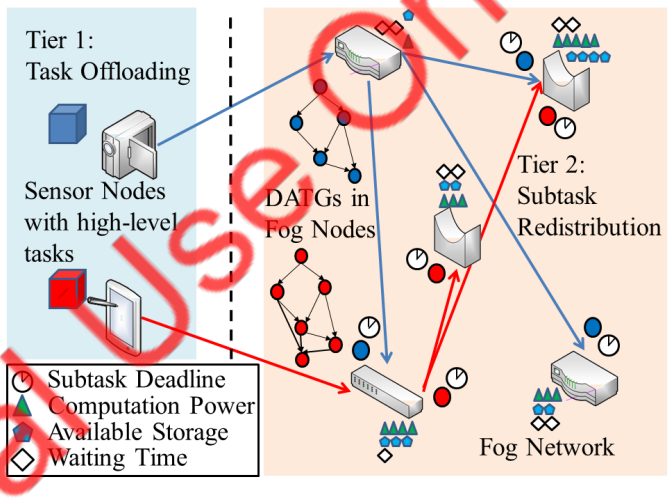


Figure 1: Proposed D2CIT scheme, where an SN offloads its task to an FN in its vicinity, and the FN redistributes the subtasks among other FNs.

To reduce operational latencies, we exploit the dependencies on the DATGs and execute the subtasks in parallel whenever possible. As shown in Fig. 1, the SNs in Tier 1 collect data corresponding to their operations. To cope with the resource-constrained nature, we formulate a greedy scheme that selects an FN for offloading the SN's task to the fog layer (Tier 2). The greedy formulation considers the energy, processing, distance, and communication delay from the SNs. We envision that this FN selects the optimized set of FNs that execute the subtasks. Due to the reasons mentioned

S. Misra and P. K. Deb are with the Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, India. e-mail: (sudipm,pallv.deb)@iitkgp.ac.in
S. P. Rachuri is with the Department of Computer Science, Stony Brook University, New York. e-mail:srachuri@cs.stonybrook.edu
A. Mukherjee is with the Department of Engineering, Cambridge University, United Kingdom. e-mail: anandarupmukherjee@ieee.org

above, we presume that the initial selection of the FN need not be an optimized choice, which is the main reason for using a greedy solution. Towards selection of the set of FNs that perform parallel execution of the subtasks, we formulate an ϵ-greedy non-stationary multiarmed bandit (MAB)-based task offloading scheme. The reason behind using a learning-based approach is the dynamic nature of mobile computing environments. The ϵ in MAB refers to the degree of exploration by an agent, which further helps in making dynamic decisions. The conditions of the devices (FNs) such as its configurations, connection links, and others vary with time due to factors such as incoming requests, queue length, processing power, and availability of cores, among others. Additionally, we also aim to achieve execution within deadlines. Online solutions such as reinforcement learning prove beneficial in overcoming the challenges in such dynamic environments. As parallelism in the fog layer also mandates transmission of the subtasks among the FNs, we anticipate tradeoffs between transmission and processing delays. However, learning-based algorithms are capable of tackling such tradeoffs and offer decisions that correspond to the local maxima of the reward function. In this work, we incorporate the difference between the deadline and the actual execution time in the reward function explicitly to penalize the FNs that exceed the deadlines. In the rest of this paper, we refer to our proposed scheme as Decentralized Distribution and Computation of Indivisible Tasks (D2CIT).

1.1 Motivation

SNs in IoT environments have hard real-time tasks that need execution within deadlines. In such scenarios, offloading the tasks to fog and edge computing platforms helps in accommodating seamless execution within deadlines [3]. These tasks involve data from a myriad of sensors and need different interdependent operations, and they need to maintain defined Quality of Service (QoS) [4]. However, most solutions focus on offloading the tasks to a single FN for sequential execution. In such schemes, only those FNs that are within the communication range of the respective SNs get utilized, while the rest remain idle, causing underutilization of available resources. However, breaking a high-level task into smaller subtasks and creating a DATG opens scope for executing the subtasks in parallel. Additionally, automating the decision at the fog layer for selecting a set of FNs for executing the subtasks helps in using idle FNs. Such automation at the fog layer also reduces the operational load in the SNs, and the fusion of parallelism helps in further reducing the latencies, which acts as the main motivation for our work. Towards this, we design a two-tier scheme that first allows an SN to select an FN (within communication range) using a greedy approach at the first tier. The greedy scheme helps in avoiding optimization time at the user level. Then, as the changing conditions of the FNs in an IoT environment mandates the need for automated learning algorithms, we formulate an Epsilon(ϵ)-greedy non-stationary MAB-based solution for selecting a set of FNs among others for executing the subtasks.

1.2 Contribution

In this work, with the assumption that the FNs form DATGs from the high-level tasks [5], we propose a decentralized task distribution system for executing simultaneously according to the levels in the DATG. The specific set of contributions in this work as follows:

- ***Utility Functions:*** We formulate utility functions for 1) determining the initial fog node (greedy approach), 2) subsequent fog nodes for executing subtasks, and 3) final reward on completion of the tasks within deadlines.
- ***Reinforcement Learning (RL):*** We formulate an ϵ-greedy non-stationary MAB-based RL solution for automating the optimized selection of FNs in such dynamic environments. The ϵ gives us the flexibility to explore FNs that are previously not chosen, opening scope for improvement in subsequent iterations.
- ***Deadlines:*** While most of the existing literature usually focuses on just reducing operational latencies, we focus on completing the executions within tolerable deadlines.
- ***Evaluation:*** Through extensive simulations, we present the feasibility and advantages of D2CIT compared to existing state-of-the-art solutions.

2 Related Work

Computational offloading to cloud and fog has been a promising solution for load balancing in dynamic environments. There has been significant efforts in this area of research. We now discuss some of the work done towards addressing this issue.

Fog computing schemes are expected to complement the services of the cloud. However, the user nodes need to decide whether to offload their tasks to the cloud or at the fog nodes. Towards this, Misra and Saha [6] proposed a scheme for deciding whether to offload to the cloud or process locally. They also proposed an optimal path selection to reach a fog node in multi-hop networks. Their greedy solution takes delays, energy consumption and dynamic network conditions into account while making the decision. Similarly, the authors in [7] formulated a game theoretic scheme for a system to choose whether to offload to the nearby devices or to the edge computing platform. Similarly, Nassar and Yilmaz proposed a resource allocation scheme in Fog Radio Access Network (Fog RAN) using reinforcement learning and Markov Decision Process (MDP) [8]. They considered the channel utility and state of the nodes as factors for deciding whether to offload to the cloud or to the fog. Researchers have also been trying to design other efficient schemes (single/multi-tier) for resource allocation in the fog layer [9]. Additionally, Azimi *et al.* [10] illustrated the scope of fog computing in healthcare applications by designing a hierarchical computing architecture to tackle the low reliability and availablity problems of the cloud. Li *et al.* [11] proposed a network architecture for vehicles in smart cities for reducing network congestion. They formulated a joint optimization scheme concerning network, cache, and computing resources.

Sarkar and Misra [12] theoretically modelled and analysed fog computing, and illustrated how it reduces latency as compared to cloud computing. Researchers have been continuosly trying to maximize the Quality of Service of the fog nodes to better serve the users [13], [14]. Task allocation among the fog nodes is analogous to the mathing problem. Concerning to this, Chiti *et al.* [15] proposed a scheme for offloading. They proposed a decentralized algorithm to select the most suitable fog node based on transmission, waiting and computation times. However, the fog node are capable of performing multiple tasks simultaneously. The authors in [16] studied to perform task scheduling and task image (task data) placement simultaneously, which aimed to jointly optimize both of them in order to minimize the maximum completion time. Researchers have also exploited the fog computing paradigm for developing energy efficient solutions [17], [18]. On the other hand, Mishra *et al* [19] proposed and implemented Particle Swarm Optimization (PSO), Binary Particle Swarm Optimization (BPSO) and Bat algorithms for service allocation in a heterogeneous fog computing environment using Virtual Machines. The authors in [20] proposed an RL method with emphasis on energy harvesting in Mobile Edge Computing (MEC) environments.

Synthesis: Task and resource allocation in cloud and fog computing is a mature field of study among researchers. As mentioned above, most of the work on fog computing relies on a single centralized agent to take the decision of FN selection. However, there is limited work on fog computing that uses DATGs and solves the problem of task offloading in a decentralized manner in a heterogeneous environment. In this paper, we propose a Reinforcement Learning based decentralized horizontal scheme to offload and distribute subtasks among FNs.

3 Problem Description

In this section, we describe our problem scenario and explain in detail the two-tier computation offloading scheme along with our set of assumptions. As shown in Fig. 1, we consider two types of devices in our scenario which are described as follows:

1) **Fog Nodes (FNs):** These are the set of heterogeneous devices that are present at the edge of the network. In addition to the networking services, these FNs also offer computation services to the requests coming from the users. In our simulation environment, we consider the configurations of gateways, hubs, switches, bridges, and routers as FNs. In IoT environments, in addition to the networking devices, RSUs also act as FNs.
2) **IoT Sensor Nodes (SNs):** These are the set of heterogeneous IoT devices that capture data for processing. In our scenario, we consider that these devices have a combination of sensors such as cameras, GPS, InfraRed, and others. The fog network acts as the external platform for the SNs for offloading their high-level tasks.

Assumption: In this work, we make the assumption that the SNs and FNs are in a quasi-static state to one another, which allows us to return the result to the SN from the FN chosen using the greedy-based solution. This does not totally restrict the movement of the SNs. In the best case, WiFi offers a communication range of 50 meters, which gives us a maximum travel distance of 100 meters (diameter). On an average, we get a travel distance of 30-50 meters (one sided radius), which is sufficient to get back the results from the fog nodes. However, in the worst case, the end devices may move out of the network, resulting in loss of communications. We plan to account for such situations and extend this work in the future.

3.1 System Model

In our system, we consider $F = \{f_1, f_2, f_3, ..., f_{\mathcal{M}}\}$ and $V = \{v_1, v_2, v_3, ..., v_{\mathcal{N}}\}$ as the sets of heterogeous FNs and SNs in a fog-enabled IoT network. The SNs communicate with the FNs wirelessly while the FNs communicate among one another via both wired and wireless means. The FNs have the ability to perform networking as well as computation operations simultaneously and they broadcast periodic beacons containing information regarding their computation power, available storage, geographical position, and the number of cores occupied in executing other requests. We represent the tasks in the SNs using three tuples as $\mathcal{T}_n =< \mathcal{T}_n^{size}, \mathcal{T}_n^{cycle}, \mathcal{T}_n^{limit} >$, such that $\mathcal{T}_n^{size}$ is the size of the task, $\mathcal{T}_n^{cycle}$ is the number of cycles required for execution, and $\mathcal{T}_n^{limit}$ is the deadline for the n^{th} FN respectively. We now define the parameters involved in this work as follows. *Computation power* $(C_x^{norm}(t))$ is the normalized number of cycles the FN/SN execute per second and is calculated as the ratio of the device's current computation power $(C_x^{dev}(t))$ and its maximum capacity (C_x^{max}),represented as $C_x^{norm}(t) = \lceil C_x^{dev}(t)/C_x^{max} \rceil$. We use x to represent the devices for the parameters which are common for both SNs and FNs, and t to represent the time instant. *Available storage* $(S_x^{norm}(t))$ is the normalized available storage space $(S_x^{dev}(t))$, calculated as the ratio of current available storage and the maximum storage capacity (S_x^{max}), represented as $S_x^{norm}(t) = \lceil S_x^{dev}(t)/S_x^{max} \rceil$. *Distance* $(d_{f_i,x}^{norm}(t))$ is the distance of i^{th} FN f_i from device x calculated using the standard distance formula on geolocations obtained from the beacon signals explained above. *Delay* $(T_{f_i}^{norm}(t))$ is the duration for which the requesting device has to wait until it receives the result of the task $(\mathcal{T})$ offloaded to i^{th} FN (f_i). For n^{th} task as time instant (t) in device x, we calculate the delay as the sum of sending time $(T_n^{send}(t))$, queueing time$(T^{queue}(t))$, and processing time $(T_{n,x}^{ps}(t))$, respectively. We calculate each of them as: $(T_n^{send}(t) = \lceil \mathcal{T}_n^{size}(t)/DataRate(t) \rceil)$, $(T_{n,x}^{ps}(t) = \lceil \mathcal{T}_n^{cycle}(t)/C_x^{dev}(t) \rceil)$, and $T^{queue}(t) = 1/(\mu - \lambda)$, where μ and λ are the service and arrival rates, respectively with each following a Poisson distribution [21], and the time required for sending the result back divided by the deadline $(\mathcal{T}_n^{limit}(t))$. Since the size of the result is miniscule compared to that of the original task, we discard it. Thus, $T_{f_i}^{norm}(t) = (T_n^{send}(t) + T_{n,x}^{ps}(t) + T^{queue})/\mathcal{T}_n^{limit}(t)$. We acquire the $DataRate(t)$ using the Shannon's formula

$DataRate(t) = BW(t) \times log(1 + SINR(t))$, where $BW(t)$ and $SINR(t)$ are the bandwidth and Signal to Interference plus Noise Ratio at time instant t, respectively. The utility function for the j^{th} SN with task $\mathcal{T}$ to choose an FN within its vicinity considering the $C^{norm}_{f_k}(t)$, $S^{norm}_{f_k}(t)$, $d^{norm}_{f_k,v_j}(t)$, and $T^{norm}_{f_k}(t)$ of the k^{th} FN as:

$$\mathcal{U}^{\mathcal{T}}_{v_j}(t) = \max_{k=1,\ldots,p} \left[\frac{C^{norm}_{f_k}(t) S^{norm}_{f_k}(t)}{(d^{norm}_{f_k,v_j}(t) + \psi) T^{norm}_{f_k}(t)} \right] \quad (1)$$

where p is the number of FNs in j^{th} SN's vicinity. In this work, the SNs choose their corresponding FN using a greedy selection based on equation 1. We describe the greedy solution in Algorithm 1. We consider each of the parameters to be a function of time to keep track of the dynamic nature of the IoT environments. For the ease of representation, we remove *(t)* from the rest of the equations.

Algorithm 1 Selection of FN f_{greedy} by SN v_j

INPUTS:
- $C_{f_i}(t)$: Computation Power of FN f_i.
- $T_{f_i}(t)$: Expected delay from FN f_i.
- $S_{f_i}(t)$: Available Storage in FN f_i.
- $d_{f_i,v_j}(t)$: Distance between FN f_i and SN v_j

OUTPUT: f_{greedy} : FN f with maximum utility function.

for i = 1 to p **do** ▷ p : No. of FNs in vicinity of v_j
  Compute $\mathcal{U}^{\mathcal{T}}_{v_j}(t)$ according to Equation (1))
  if $\mathcal{U}^{\mathcal{T}}_{v_j,f_i}(t) \geq \mathcal{U}^{\mathcal{T}}_{v_j,max}$ **then**
    Set $f_{greedy} = f_i$
    Update $\mathcal{U}^{\mathcal{T}}_{v_j,max}$
if $p == 0$ **then** ▷ If v_j has no FN within its vicinity
  Perform computation locally

3.2 D2CIT: Decentralized Distribution and Computation of Indivisible Tasks

As mentioned earlier, the dynamic nature of IoT environments mandates the need for automated solutions. In our scenario, the states of FNs keep changing as the SN requests for computational service increases. Towards this, once the SN sends its task to an FN, we design an ϵ-Greedy Non-Stationary MAB-based solution to select the set of FNs for executing the subtasks. This allows us to exploit the services of FNs that have better performance and also explore other FNs with probability ϵ. Also, to reduce our search space, we only consider FNs that have $T_x(t) \leq \mathcal{T}^{limit}_n$ and have at least one core available to serve a request. Thus, our approach consists of three parts, namely *Agent*, *Arms*, and *Rewards*. In our scenario, *Agent* is the FN f_{greedy} that is chosen by an SN based on the greedy choice and *Arms* are the FNs in the scope list. The *Agent* tries to gain knowledge about the *Arms* and exploit them to make an optimized decision while maximizing the reward. The decision to select arms leads to a reward according to the probability distribution of the concerned arms. In the case of non-stationary MAB, the rewards keep changing with time, which is analogous to the changes in the performance of an FN. For calculating the reward, the FN f_{greedy} first identifies the FNs that have the potential to operate on the subtasks within deadline ($\mathcal{T}^{limit}_{subtask}$). We then measure the time taken by the FNs and in then reward based on how early the subtasks got completed.

3.3 Reward

We express the rewards for the FN f_{greedy}, we consider how fast the chosen set of FNs completes the subtasks. On completion of the whole task, $\mathcal{T}^{limit}_{subtask} - T_x$ expresses the speed of execution concerning the deadline. Additionally, we also consider the size of the subtasks offloaded. It is undesirable to offload tasks to long distances as well as consumption of power $\mathcal{P}^{offload}_{f_{greedy}}$. We formulate a utility function for an FN on assigning a subtask and inputs to the reward table as:

$$\mathcal{U}^{subtask}_{f_{greedy}} = \frac{(\mathcal{T}^{limit}_{subtask} - T_x)\mathcal{T}^{size}_{subtask}}{(d_{f_k,v_j} + \psi)\mathcal{P}^{offload}_{f_{greedy}}} \quad (2)$$

In order to limit the rewards to the range of $(0, 1)$, we use the sigmoid function as $\mathcal{R}^{subtask}_{f_i} = e^{\mathcal{U}^{subtask}_{f_{greedy}}}(1 + e^{\mathcal{U}^{subtask}_{f_{greedy}}})^{-1}$ Finally, we calculate the reward for f_{greedy} as:

$$\mathcal{R}^{subtask}_{f_i} = \frac{e^{\frac{(\mathcal{T}^{limit}_{subtask} - T_x)\mathcal{T}^{size}_{subtask}}{(d_{f_k,v_j} + \psi)\mathcal{P}^{offload}_{f_{greedy}}}}}{1 + e^{\frac{(\mathcal{T}^{limit}_{subtask} - T_x)\mathcal{T}^{size}_{subtask}}{(d_{f_k,v_j} + \psi)\mathcal{P}^{offload}_{f_{greedy}}}}} \quad (3)$$

Every FN maintains a table that consists of rewards that it has obtained on offloading to every other FN. The dynamic nature of the IoT environments often forces the MAB strategy to make wrong decisions on using standard averaging as the estimation function. Instead, for FN f_i, we compute the reward for f_{greedy} as a weighted average of rewards with reduced weights for older rewards. Although the changes in IoT environments are continuous, the correlation exists more towards the recent past. We express this using the weighted average mentioned earlier, which helps in discarding scenarios in the past that have no connection to the current one. Thus, we calculate the average reward for executing the whole task as:

$$\mathcal{R}^{avg}_{f_i} = \frac{\mathcal{R}^n_{f_i} + \rho\mathcal{R}^{n-1}_{f_i} + \rho^2\mathcal{R}^{n-2}_{f_i} + \ldots + \rho^n\mathcal{R}^0_{f_i}}{1 + \rho + \rho^2 + \ldots + \rho^n} \quad (4)$$

where $\rho < 1$ is the step size, and $\mathcal{R}^y_{f_i}$ is y^{th} reward obtained. With (ϵ)-greedy strategy, *Agents* explore the environment by randomly choosing the arm with the probability of (ϵ) and exploit the knowledge of rewards by greedily choosing the *Arm* with the highest output of estimation function with the probability $(1 - \epsilon)$. We summarize our MAB scheme in Algorithm 2.

4 Performance Evaluation

In our scenario, we consider multiple SNs which need to offload high-level tasks, and multiple FNs connected by strong backhaul network (Fog Network) that cooperate among one another in providing computational services.

Algorithm 2 Selection of FN f_k by FN f_{greedy}

INPUTS:
- Cores left in f_i
- $C_{f_i}(t)$: Computation Power of f_i
- $d_{f_i,f_{greedy}}$: Distance between f_i and f_{greedy}
- $\mathcal{T}_n$: Subtask Tuple
- Rewards Table of f_{greedy}

OUTPUT: f_k : FN f with highest Avg Reward

for i = 1 to $\mathcal{M}$ **do** ▷ $\mathcal{M}$: Total No. of FNs
  Compute *Delay* $T_{f_i}(t)$ using $\mathcal{T}_n$
  if $T_{f_i}(t) \leq \mathcal{T}_n^{limit}$ and Cores left in $f_i \geq 1$ **then**
    Add f_i to scope list
if length of scope list is 0 **then**
  Perform computation locally
else
  for i = 1 to $\mathcal{S}$ **do** ▷ $\mathcal{S}$: No. of FNs in scope list
    for each subtask in $\mathcal{T}_n$ **do**
      Compute $\mathcal{R}_{f_i}^{avg}$ according to Equation (4)
      if $\mathcal{R}_{f_i}^{avg} \geq \mathcal{R}_{max}^{avg}$ **then**
        Set $f_k = f_i$
        Update $\mathcal{R}_{max}^{avg}$

Table 1: Simulation Parameters

Parameter	Value
SN computation capability	$0.1, 0.2, 0.3\ GHz$
FN computation capability	$0.8, 1.0, 1.5\ GHz$
Cloud computation capability	$3.9\ GHz$
Task arrival Rate at FN	$188.391, 299.985, 427.429\ MIPS$
Task size	$420 - 12600\ KB$
Corresponding Cycles required by tasks	$1000 - 30000$
Shannon Capacity	$30\ KBps$
Energy to transfer 1B data	$20\ nJ$
FN Power for processing	$2\ W$

The SNs have a minimal range for communication as they communicate wirelessly with the FNs. We consider static positions for the FNs. On the other hand, we assume that the SNs will remain in the FNs vicinity during the execution of the tasks and hence are quasi-static to one another.

In this section, we present the results illustrated by D2CIT during our simulations. Motivated by the simulation parameters in [12], we run our experiments on those listed in Table 1. We randomly deploy the FNs and the SNs on a square area of $1000 \times 1000\ m^2$ with a cloud server at the center. The FNs and SNs are capable of WiFi communications due to which we set the communication range as 30 – 50 meters, respectively. We compose our code in Python and execute on an i5 processor for running our simulations. Upon implementing reinforcement learning algorithms, the FNs receive a reward for selecting the correct subset of FNs for performing the sub-task executions. Since most of the existing solutions focus on offloading the tasks to a single FN in its entirety, we compare our results with generalised cloud and fog computing schemes and define them as follows:

1) *Cloud Computing:* We place the cloud at the center of our simulation area and offload tasks from the SNs. Since the resources in these servers are virtually unlimited, we do not impose any restrictions on the number of incoming requests.

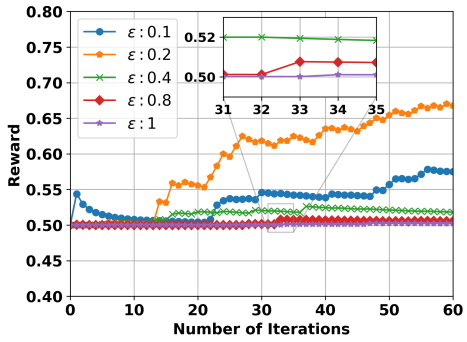
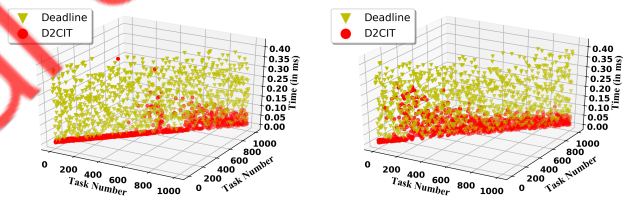


Figure 2: Rewards for the FNs for redistributing the subtasks and executing the tasks within deadlines with varying ϵ values.

2) *Fog Computing:* We consider the same FN configurations and positions as in D2CIT for the fog computing scheme. However, instead of redistributing the subtasks among the other FNs, we execute the tasks sequentially in its entirety. Additionally, as we consider our FNs to have 4 cores, we perform tasks from 4 SNs simultaneously.

We now present the rewards received by an arbitrary FN during one of our experiment iterations.



(a) 100 Fog Nodes (b) 500 Fog Nodes

Figure 3: Comparison of the deadlines set and the actual execution time of the tasks illustrated by D2CIT.

4.1 Rewards

We present the rewards from an FN during an experiment running with 1000 SNs and 500 FNs. We vary the ϵ (exploratory) factor to understand how the performance varies. In Fig. 2, we observe that the system works best when $\epsilon = 0.2$. We observe a similar trend at 0.1. However, the reward value is significantly low. From such trends, we establish that some amount of variability in FN selection is beneficial. On setting higher ϵ, we observe a sudden drop in the reward values. Intuitively, this is because, in a set of fixed FNs, the FN performing the selection already chooses the best subset. Further changes lead to the selection of less efficient FNs, which causes deterioration in the outcome. Thus, for the rest of our simulations, we set $\epsilon = 0.2$ and observe the results.

4.2 Deadlines

We demonstrate how D2CIT performs with respect to the deadlines set for each subtask. In our experiments, we set a

range of deadlines and observe the results. In Figs. 3(a) and 3(b), we observe that majority of the tasks finish execution within the deadlines. Although we notice a few tasks in each of the scenarios that have higher delays as compared to the others, they do not exceed the thresholds. We believe that the higher delays are due to congestion and increased waiting time as the number of requests increases. In the future, we plan to address this issue by developing schemes that changes the chosen subset of FNs. Since all of the tasks finish execution within the deadlines, we conclude that our proposed D2CIT scheme performs within tolerance level with respect to each task in a fair manner.

4.3 Latencies

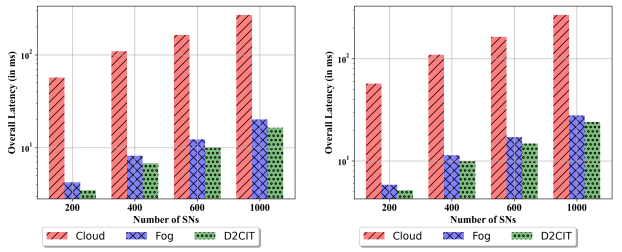
We present the latencies in case of D2CIT and compare it with that of general Cloud and Fog Computing schemes. We show the overall latency and then present a breakdown of the latencies for transmission and processing. In the following results, we fix the number of FNs (100 and 500) and vary the SNs from 200 – 1000 in each case. For better representation, we present each of the latencies in log scale.

4.3.1 *Overall Latency*

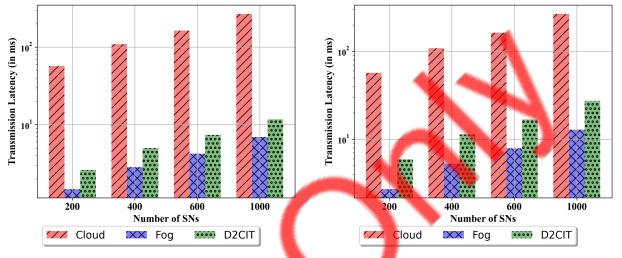
The overall latency is inclusive of the time required to send the tasks to the FNs until reception of the results and present them in Figs. 4(a) and 4(b). Due to the introduction of parallelism, we observe that in all cases, the latencies by D2CIT is much lower than the general Cloud and Fog Computing schemes, which is the main motive of our work. In particular, D2CIT offers a reduction of 17% in the overall delay. This is because the proposed method explores the availability of other FNs (not best-effort) which may offer better delays in addition to the parallel computations. As the number of SNs increases, the number of tasks also increases, which again significantly increases the number of subtasks, which increases the waiting time in FNs proportionately. Due to this increase in the waiting time, we observe an increasing trend in the latencies. However, the ratio of the difference in the general schemes and our proposed method remains the same.

4.3.2 *Transmission Latency*

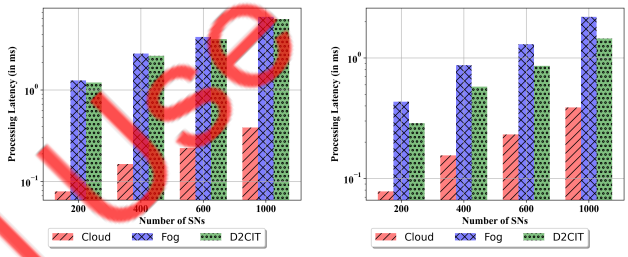
Transmission latencies includes the time invested in transferring tasks and subtasks from the SNs to the FN (initial selection) and then to the other FNs. In Figs. 4(c) and 4(d) we observe that the transmission latency is maximum in case of Cloud computing schemes, while that in case of Fog computing, it is minimum. As per our expectation, the latency is maximum in case of D2CIT. The increase latency is because of the further transmission of the subtasks among the FNs in the fog layer. We exclude the time for returning back the result to the SN as it is usually a single packet which is relatively smaller than the original tasks. Due to this minuscule size, we neglect the time concerning the transfer of the result packets.



(a) 100 Fog Nodes (b) 500 Fog Nodes

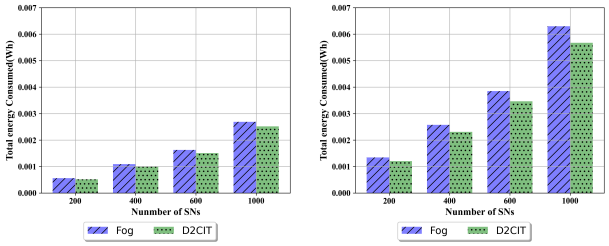


(c) 100 Fog Nodes (d) 500 Fog Nodes



(e) 100 Fog Nodes (f) 500 Fog Nodes

Figure 4: Comparison of latencies endured by SNs in case of general fog, cloud, and D2CIT schemes with varying number of SNs and fixed number of FNs.



(a) 100 Fog Nodes (b) 500 Fog Nodes

Figure 5: Comparison of energy consumption by general fog computing schemes and D2CIT with varying number of SNs.

4.3.3 *Processing Latency*

Processing latencies are the time corresponding to the summation of the execution time of the subtasks among the FNs. Due to the high configurations of cloud servers, the processing time is minimum. On the other hand, as the FNs in general Fog Computing schemes perform each of the subtasks sequentially, we observe in Figs. 4(e) and 4(f) that D2CIT offers lower latencies than the former. Further, we notice that as the number of FNs increases, the latencies

decrease further. This decrease in latency is due to the availability of idle FNs, which have the potential to provide service on demand. However, the FNs might be at distant locations, which increases the transition latency, as shown in Section 4.3.2. Although we conclude that our proposed D2CIT scheme scales well, we strongly feel the need for an optimized relation to determining the number of FNs for the corresponding number of SNs in the future.

4.4 Energy Consumption

We compare the energy consumption in the case of D2CIT and compare its results with the general fog computing solutions. We calculate the energy consumption based on the metrics in Table 1. We observe in Figs. 5(a) and 5(b) that in all the scenarios, D2CIT demonstrates reduced energy requirement. Additionally, we notice that with the increasing number of FNs, D2CIT further reduces energy consumption, which further increases the difference between the two schemes. We attribute this to the ϵ in the MAB solution, which allows D2CIT to explore a range of FNs that offer better performance than those present closer or than those selected earlier. However, with the increasing number of FNs, the energy consumption increases, which is intuitive as more number of devices needs more energy to operate.

4.5 Benchmark Solution

Existing literature usually does not focus on performing tasks within deadlines and also does not focus on breaking the tasks into smaller subtasks to exploit parallelism. However, researchers have been developing new methodologies for improving computation offloading in fog enabled environments. To demonstrate the performance efficiency of D2CIT, we compare our results with another computation offloading scheme named BLOT [22] by Zhu *et al.* Authors in this work formulated a bandit learning algorithm for determining FNs online. However, they only offload the task to the FNs in its entirety. We present a comparative study of the processing latencies and speedup of BLOT with our proposed scheme.

4.5.1 Latency

We run simulations of D2CIT as well as BLOT in an area of $1000 \times 1000\text{m}^2$ containing 1000 SNs with varying FNs $(100, 250,$ and $500)$. We observe in Fig. 6 that although BLOT also performs the tasks within deadlines, the average processing time is higher than that of D2CIT. Additionally, in every scenario, the performance of BLOT remains the same throughout. On the other hand, the delays in D2CIT gradually decreases as the number of FNs increases. We believe that since D2CIT utilizes idle FNs and avoids those with high waiting times, which leads to a reduction in the processing latency. We conclude that D2CIT scales well, and witness the natural intuition that latency decreases as the number of FNs increases.

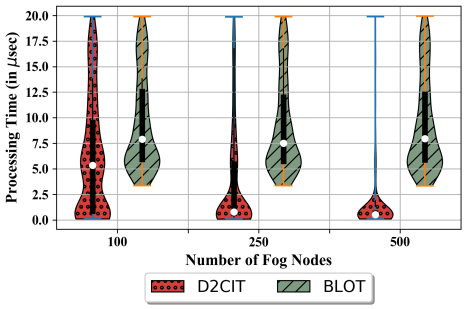


Figure 6: Distribution of delays on comparing D2CIT with existing solution.

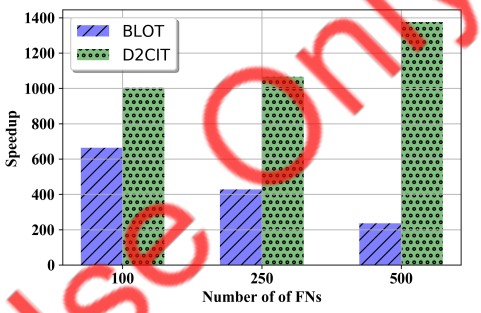


Figure 7: Speedup in case of D2CIT and existing solution.

4.5.2 Speedup

We calculate the speedup as the ratio of the time required for executing the tasks using D2CIT/BLOT to the time required for sequential execution and is a unitless factor. We observe in Fig. 6 that D2CIT demonstrates improved latency as compared to BLOT and that the average latency decreases further down as the number of FNs increases. Consequently, we notice in Fig. 7 that with increasing FNs, the speedup value of BLOT decreases while that in the case of D2CIT increases. On average, in comparison to BLOT, we observe a 59% increase in speedup on implementing D2CIT. Intuitively, the introduction of parallelism in D2CIT plays a significant role in obtaining such increased speedups. The FNs have to perform very small subtasks, which makes them available for other tasks instantaneously, which is not possible in conventional methods. We observe a decreasing trend in the case of BLOT on increasing number of FNs. This is because the computing fog nodes under this scheme share their execution overheads only by offloading to other devices that have better configurations. Taking the constraint of executing tasks within deadlines, BLOT chooses fog nodes that are computationally superior but induces additional delays due to data forwarding and subsequent transmissions on adding more FNs. We infer that D2CIT significantly increases resource utilization in the FNs.

In summary, we observe the following from our simulations:

- The reward function in equation 3 tries to maximize the utility and the exploration factor plays a major role. We observe that the proposed model works best for $\epsilon = 0.2$ (as shown in Fig. 2) in Section 4.1.

- In Sections 4.2, 4.3, and 4.4, we observe reduced delays and energy consumptions in comparison with traditional (best-effort) fog and cloud computing schemes while executing the tasks within deadlines.
- In comparison to state-of-the-art solutions, we observe that D2CIT offers reduced latencies and increased speedup in Section 4.5.

5 Conclusion

In this work, we presented a two-tier distributed scheme D2CIT, for computation offloading in a fog-enabled IoT environment. We considered SNs with tasks with deadlines that have computational needs. We envision D2CIT to break down the high-level tasks to smaller subtasks and form a DATG. We design a greedy solution for the selection of the FN, which breaks down the task from the corresponding SNs. For automating redistribution of the subtasks in a dynamic environment, we design an Epsilon(ϵ)-greedy non-stationary multiarmed bandit-based scheme. To illustrate the operational efficiency of D2CIT, we presented the deadlines, latency, and energy consumption from our simulation results. Additionally, we also compared D2CIT with existing solutions and presented how our proposed work performs better concerning latency and speedup.

In the future, we plan to extend this work by accounting for issues due to mobility and loss of communications. Towards this, we plan to return the results to the SNs from the nearest FN than the initially chosen FN using the greedy solution.

References

[1] X. Hou, Y. Li, M. Chen, D. Wu, D. Jin, and S. Chen, "Vehicular Fog Computing: A Viewpoint of Vehicles as the Infrastructures," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 6, pp. 3860–3873, Jun. 2016.

[2] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog Computing and Its Role in the Internet of Things," in *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, Aug. 2012, pp. 13–16. [Online]. Available: http://doi.acm.org/10.1145/2342509.2342513

[3] 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, 2017.

[4] L. T. Tan and R. Q. Hu, "Mobility-Aware Edge Caching and Computing in Vehicle Networks: A Deep Reinforcement Learning," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 11, pp. 10 190–10 203, Nov. 2018.

[5] J. Song, Q. Li, and S. Ma, "Toward Bounds on Parallel Execution Times of Task Graphs on Multicores with Memory Constraints," *IEEE Access*, vol. 7, pp. 52 778–52 789, Apr. 2019.

[6] S. Misra and N. Saha, "Detour: Dynamic Task Offloading in Software-Defined Fog for IoT Applications," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 5, pp. 1159–1166, May 2019.

[7] S. Joilo and G. Dn, "Decentralized Algorithm for Randomized Task Allocation in Fog Computing Systems," *IEEE/ACM Transactions on Networking*, vol. 27, no. 1, pp. 85–97, Feb. 2019.

[8] A. T. Nassar and Y. Yilmaz, "Reinforcement-Learning-Based Resource Allocation in Fog Radio Access Networks for Various IoT Environments," *Computing Research Repository (CoRR)*, vol. abs/1806.04582, Aug. 2018. [Online]. Available: http://arxiv.org/abs/1806.04582

[9] Q. Li, L. Zhao, J. Gao, H. Liang, L. Zhao, and X. Tang, "SMDP-Based Coordinated Virtual Machine Allocations in Cloud-Fog Computing Systems," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1977–1988, Jun. 2018.

[10] I. Azimi, A. Anzanpour, A. M. Rahmani, T. Pahikkala, M. Levorato, P. Liljeberg, and N. Dutt, "HiCH: Hierarchical Fog-Assisted Computing Architecture for Healthcare IoT," *ACM Trans. Embed. Comput. Syst.*, vol. 16, no. 5s, pp. 174:1–174:20, Sep. 2017. [Online]. Available: http://doi.acm.org/10.1145/3126501

[11] M. Li, P. Si, and Y. Zhang, "Delay-Tolerant Data Traffic to Software-Defined Vehicular Networks With Mobile Edge Computing in Smart City," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 10, pp. 9073–9086, Oct. 2018.

[12] S. Sarkar and S. Misra, "Theoretical Modelling of Fog Computing: A Green Computing Paradigm to Support IoT Applications," *IET Networks*, vol. 5, no. 2, pp. 23–29, Mar. 2016.

[13] H. Shah-Mansouri and V. W. S. 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.

[14] S. F. Abedin, M. G. R. Alam, S. M. A. Kazmi, N. H. Tran, D. Niyato, and C. S. Hong, "Resource Allocation for Ultra-Reliable and Enhanced Mobile Broadband IoT Applications in Fog Network," *IEEE Transactions on Communications*, vol. 67, no. 1, pp. 489–502, Jan. 2019.

[15] 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, Dec. 2018.

[16] D. Zeng, L. Gu, S. Guo, Z. Cheng, and S. Yu, "Joint Optimization of Task Scheduling and Image Placement in Fog Computing Supported Software-Defined Embedded System," *IEEE Transactions on Computers*, vol. 65, no. 12, pp. 3702–3712, Dec. 2016.

[17] L. Liu, Z. Chang, and X. Guo, "Socially Aware Dynamic Computation Offloading Scheme for Fog Computing System With Energy Harvesting Devices," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1869–1879, Jun. 2018.

[18] A. Boukerche and P. Sun, "A Novel Hierarchical Two-Tier Node Deployment Strategy for Sustainable Wireless Sensor Networks," *IEEE Transactions on Sustainable Computing*, vol. 3, no. 4, pp. 236–247, Oct. 2018.

[19] S. K. Mishra, D. Puthal, J. J. P. C. Rodrigues, B. Sahoo, and E. Dutkiewicz, "Sustainable Service Allocation Using a Metaheuristic Technique in a Fog Server for Industrial Applications," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4497–4506, Oct. 2018.

[20] M. Min, L. Xiao, Y. Chen, P. Cheng, D. Wu, and W. Zhuang, "Learning-Based Computation Offloading for IoT Devices With Energy Harvesting," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 2, pp. 1930–1941, Jan. 2019.

[21] D. Gross, J. F. Shortle, J. M. Thompson, and C. M. Harris, *Fundamentals of Queueing Theory*, 4th ed. USA: Wiley-Interscience, 2008.

[22] Z. Zhu, T. Liu, Y. Yang, and X. Luo, "BLOT: Bandit Learning-Based Offloading of Tasks in Fog-Enabled Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 12, pp. 2636–2649, Dec. 2019.