International Journal of Computer Networks and Applications (IJCNA)
DOI: 10.22247/ijcna/2023/220739 Volume 10, Issue 2, March – April (2023)

RESEARCH ARTICLE

A Collaborative Offloading Task Framework for IoT Fog Computing

Amira S. Ibrahim
Department of Computer Science, Suez Canal University, Ismailia, Egypt.
amira_ibrahim@ci.suez.edu.eg

Hassan Al-Mahdi
Department of Computer Science, Suez Canal University, Ismailia, Egypt.
drhassanwesf@ci.suez.edu.eg

Hamed Nassar
Department of Computer Science, Suez Canal University, Ismailia, Egypt.
nassar@ci.suez.edu.eg

Received: 23 January 2023 / Revised: 04 March 2023 / Accepted: 10 March 2023 / Published: 29 April 2023

Abstract – Fog computing is a viable approach to improving the performance of cloud computing, especially in terms of response time, which is critical to real-time applications. Specifically, the fog brings the cloud resources closer to terminal devices (TDs), thereby decreasing latency and increasing throughput. The problem of task offloading from TDs to the fog has enjoyed much research work, but the issue of TD mobility has not found enough attention, and hence is the present work. Herein, TD mobility in fog computing involves the transfer of services while the TD is moving from one fog to another, requiring delicate coordination between the fogs. To this end, a framework is proposed to ensure that the fogs together with the cloud collaborate, first to always keep track of the current location of the TD offloading the task, and second to accurately serve the task in a distributed fashion while the TD is moving. The framework dedicates two queues in each fog, one to receive fresh tasks from TDs and one to receive hand-over tasks from other fogs, and leverages a vigilant inter-fog messaging system capable of keeping all concerned components abreast of the latest status. A program has been written in Python to simulate the framework and example operational environments. The program has been used to perform extensive experiments in order to assess the performance of the framework under high and low mobility conditions. The findings indicate that the framework is highly reliable and can deliver, under various mobility modes, the right response to the right TD at the right time.

Index Terms – Internet of Things, Task Offloading, Response Time, Mobility, Cloud Computing, Fog Computing.

1. INTRODUCTION

Recently, the Internet of Things (IoT) has become one of the most important paradigms that affect the technology industry. The CISCO Annual Internet Report (AIR) released in 2020 [1] indicates that there will be about 29.3 billion network devices in the World by the end of 2023. Juniper Research [2] has listed that in 2022 there will be about 50 billion IoT devices. The international data corporation (IDC) predicting that the number connected devices worldwide will reach 55.7 billion devices, with about 75% of them IoT devices. Besides, the IDC predicted that the connected IoT devices will produce 73.1 ZB of data by 2025, up from 18.3 ZB in 2019.

A great variety of delay-sensitive tasks are generated by IoT devices. Due to their resource constraints, IoT devices have to offload their tasks to cloud servers for faster processing [3].

The Cloud Computing paradigm has been established to provide a set of centralized cloud servers with more computational and storage capabilities [4]. Due to its flexible and scalable design, the cloud has been used to process tasks offloaded by IoT devices [5]. However, due to the centralization and remote location of cloud servers, there are numerous problems related to such issues as bandwidth, device mobility and task response time, which need careful consideration [6].

To mitigate these problems, the concept of Fog computing was presented by CISCO to bring the power of cloud computing (both resources and services) closer to IoT devices at the edge of the network, forming a fog layer [7], [8]. As depicted in Figure 1, the fog layer consists of a set of fog nodes (FNs), where each FN is usually associated with a network access node, e.g. base station (BS) [9].

Due to the fog layer being close, more networking, storage and computing functions are available in a more convenient way for IoT devices [10], [11]. As a result, Fog computing could be more efficient than cloud computing to serve IoT

RESEARCH ARTICLE

application requirements through task offloading, which is a function that enables IoT devices to transmit a part of their tasks to a selected FN that may overcome the problems of latency and energy consumption [12]. As IoT devices are co-located under the coverage of a BS, a massive number of tasks are received at the FN. This results in scalability challenges because of the constrained processing capacity of the FN. To reduce the load of static FNs, an opportunistic FN (OFN) solution was introduced in [13]. In this solution, when an IoT device is close to an OFN, the former can offload a task to the latter and receive the response directly and quickly. This draws attention to the notion of IoT device mobility.

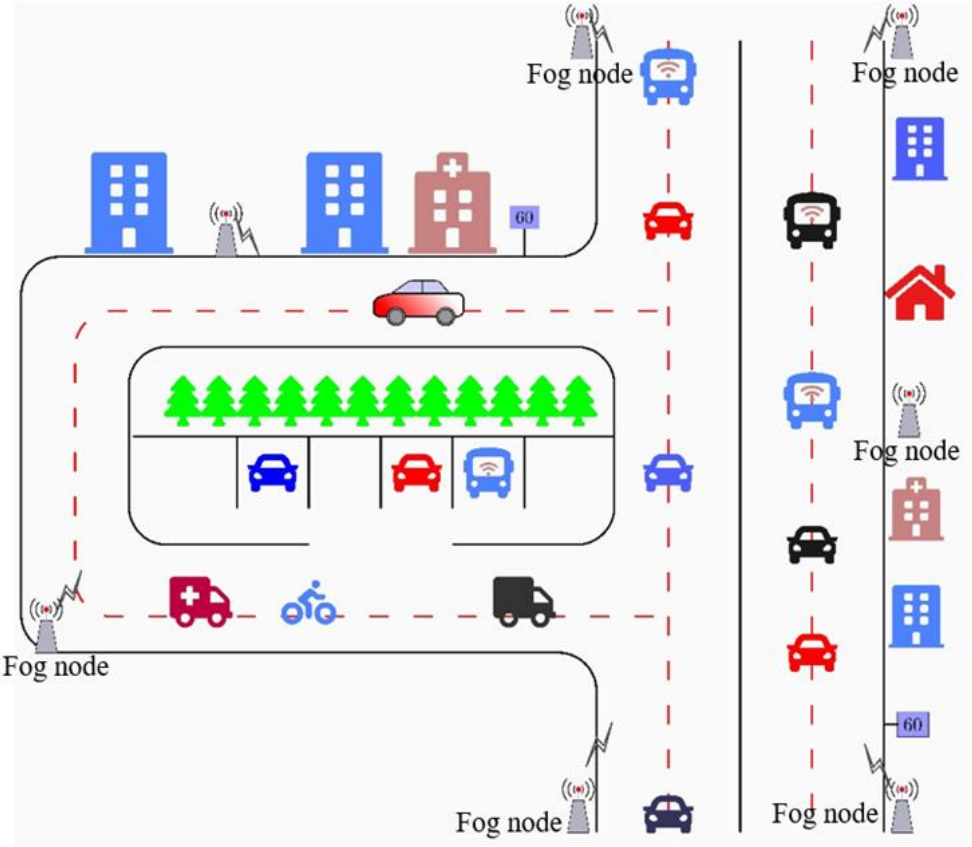

Figure 1 Fog Computing Network Conceptual Design

IoT device mobility, addressed by the present article, is an important practical consideration that has not enjoyed much research attention. The mobile device may be carried by humans, robots or unmanned aerial vehicles. Mobility of IoT devices affects performance of Fog computing, as they change access points quite frequently [14].

In particular, after the FN starts a task computation activity for a certain IoT device, the device may leave the FN to another FN. Without taking design precautions, the IoT device will retransmit the same task to the new FN, unnecessarily congesting the FNs, eating up precious bandwidth and wasting much energy. In [15], a deadline-aware and cost-effective offloading approach was introduced to improve offloading efficiency for mobile IoT devices with the assumption that the IoT devices move with high speed. A scheme was then introduced to allow IoT device make more efficient offloading decisions.

In this paper, a Collaborative Offloading Task Framework for IoT Fog Computing is proposed to handle the mobility of IoT devices more efficiently, reducing the task response time. The main idea of the framework is to migrate the task response of the mobile IoT device as the latter moves from one FN to another. That is, the response of the offloaded task may be delivered to the mobile IoT device by an FN other than the FN that originally received the task from the device. This migration distributes the processing of the task among many FSs, sparing the device the trouble of resubmitting the task to each FN it moves to, improving the QoS greatly.

1.1. Problem Definition

A serious challenge arises in fog computing when the IoT devices served by the fog system are mobile. Specifically, after a FN starts a task computation activity for a certain IoT device, the device (being mobile) may depart to another FN. Unless design measures are in place to intervene, the IoT

RESEARCH ARTICLE

device will have to retransmit the same task to the new FN. This retransmission, which effectively throws away whatever computation done in the old FN, will have bad consequences, such as congesting the FNs, eating up bandwidth, dissipating energy and wasting precious time which can possibly lead to missing a critical deadline. In this article a comprehensive framework is proposed to mitigate this challenge.

The remainder of this paper is organized as follows. In Section 2, a summary of related work is provided. In Section 3, the system model is presented and explained and the fog network architecture used in the proposed framework is outlined. The framework is illustrated in section 4, and is evaluated in section 5. Finally, concluding remarks are presented in section 6.

2. RELATED WORK

Diligent attempts have been made in recent years to investigate the issue of mobility fog computing. Raouf et al. [16] proposed a mobility-aware task offloading enhancement scheme for fog computing. This scheme was evaluated in terms of CPU cycles and time delay cost for both task offloading and local computing. Kyung et al. [5] proposed an opportunistic fog node (OFN) offloading architecture that could be dynamically flexible depending on the mobility of the OFN. An analytical model is developed considering hybrid, indirect and direct OFN offloading cases. The results indicated that the hybrid and direct offloading result in better performance than indirect offloading.

Using the concept of matching theory, Chiti et al. [17] proposed a distributed algorithm for offloading tasks to fog nodes in IoT systems, while Zhao et al. [18] introduced an offloading scheme in fog radio- access-networks (F-RANs). They both optimized offloading decisions, computational resource allocation, and the allocation of radio resource to minimize the weighted sum of the total power consumption and the total offloading latency. The problem was formulated as non-convex and solved with a non-linear and iterative algorithm that runs in polynomial time. Du et al. in [19] proposed a study aiming to reduce the maximum weighted cost of delay and power consumption in a system with mixed cloud/fog computational offloading. The authors formulated the problem as a mixed non-linear-programming (MINLP) type and solved it for the optimum offloading decision.

In [20], the authors introduced a solution to find optimum task offloading decisions and path. The problem was formulated as an integer linear programming (ILP) type and solved with a greedy heuristic-based approach. In [21], Liu et al. have proposed a multi-objective function based on power consumption, delay and payment costs to optimize the offloading decision and the transmit power. The authors used different queuing models to explore the processing behavior of the elements of the network. Yao et al. in [22] explored task offloading with the aim of reducing the cost of the system by considering the tasks QoS requirements and optimizing the number of rented virtual machines and the power consumption. In [23], the authors studied the offloading problem to minimize service latency through load balancing and fog collaboration. In [24], the authors introduced a distributed collaborative computational offloading algorithm in a game theory form model.

Ghosh et al. [25] proposed a real-time cloud fog edge IoT collaborative framework, namely Mobi-IoST, to efficiently deliver processed information to user devices based on intelligent decision making and predictions of user mobility. The framework uses agent mobility knowledge to predict user location. Based on the user location, the processed information with low latency and low power is delivered. Lakhan et al. in [15] proposed an enhanced vehicular fog cloud network scheme based on blockchain and multi-side offloading with mobility, fault-tolerance, and mobility limitations. The basic goal was to reduce real-time communication costs, subject to specific constraints like task deadlines and network bandwidth. The authors of [26], developed a programming technique that maximizes the number of successfully processed IoT tasks with adequate security criteria while minimizing the end-to-end transmission delay. To reduce task execution and power consumption in a fog, the authors of [27] suggested a task offloading technique that takes into account both communication and computing delays.

Despite the huge amount of research work published in the past years, more work on mobile fog computing is still appearing till this day. In [28], the authors devise a scheme for placing and selecting fog nodes (FN) for maximum utilization of resources in the context of the Industrial Internet of Things (IIoTs). Specifically, they propose a multi-level hierarchical deployment model using the IoT devices themselves as FNs. They consider for the selection of a device to be a FN many parameters, such as energy, path, location, storage, and available computing resources. In [29], the authors assume that the FNs are already in place and focus on how a mobile device selects the best FN in order to maximize resource utilization and throughput at the same time. They present an algorithm based on classification and regression trees, taking into account critical considerations such as authentication, confidentiality, integrity, availability, capacity, speed, and cost.

In the context of vehicular networks, the authors of [30] devise a scheme to maximize both resource utilization and throughput by adjusting the ratio of tasks to be offloaded to the FNs. In particular, they present a partial computation offloading and adaptive task scheduling algorithm with two phases. First, a two-sided matching algorithm is invoked to derive the optimal transmission scheduling discipline, then the

RESEARCH ARTICLE

offloading ratio of vehicular users is obtained through convex optimization. Also for vehicles, the authors of [31] devise a scheme to quickly process tasks offloaded by moving vehicles by FNs installed along the road side via a utility function and a knapsack-based task scheduling algorithm. Using a knapsack also are the authors of [32] who formulated with it an optimization offloading algorithm to minimize the energy consumption of the offloading mobile devices. For resource allocation and dynamic scheduling, they present a dynamic scheduling algorithm based on a priority queue. For a thorough and recent review of the modern trends to select the best vehicular FN, one can consult the survey in [33]. Although the survey is dedicated to vehicular fog computing it has a wealth of information that can be leveraged in fog computing in general.

By surveying the work on mobile fog computing, it can be easily realized that an important issue regarding the processing of a task that has been offloaded by a mobile device, frequently changing FNs, has not been properly addressed. The issue concerns the handling of a task that was offloaded in a certain FN by a certain mobile device which, due to mobility, later departs to a new FN before receiving the response from the old FN. How to handle this task collaboratively by the fog system in an optimal way that saves on energy, time and bandwidth is the focus of the present article.

3. SYSTEM MODEL

As shown in Figure 2, the considered system consists of three layers, namely Mobile computing (MC), Fog computing (FC), and Cloud computing (CC). The MC layer (aka perception layer) is the lowest level and typically has a massive number of mobile terminal devices (TDs) geographically distributed along a highway service area. The TDs can be IoT devices such as sensors, smartphones, tablets, smart watches, smart-glasses and actuators, with all generating tasks all the time. In addition, each TD has its own resources such as CPU power, battery, memory, and wireless transceiver interface card. Due to memory limitation, each TD is equipped with a small finite buffer for hosting tasks to be executed later when computing resources are available.

The FC layer (aka network edge) consists of a set of Fog nodes distributed along the highway road. Each Fog node receives offloading tasks from the TDs residing in its coverage area. Because of its proximity to the TDs, Fog nodes decrease the workload that would otherwise go to the cloud, ensuring also low latency for the TDs in its coverage area.

As shown in Figure 3, the geographic area along the highway is divided into a number N of cells, denoted by $c_1, c_2, \ldots, c_N$. It is assumed that, cell c_k with a unique identifier idc is served by only one fog node f_k with a unique identifier idf. A complete view of the state of fog nodes is obtained via a Computing server (CS) in the backbone network. The number of TDs that can be served within a given cell varies from one cell to another. Each TD i has a unique identifier idt within all cells. Furthermore, each TD i within cell c_k has a profile P_i^k made up of the triplet $\langle idf, idc, idt \rangle$. The CS has a data structure (DSC) used to store the profiles of all the TDs currently in the cells. The CS is in charge of assigning the identifier idt to each TD arriving at the network, i.e., at any cell, under consideration.

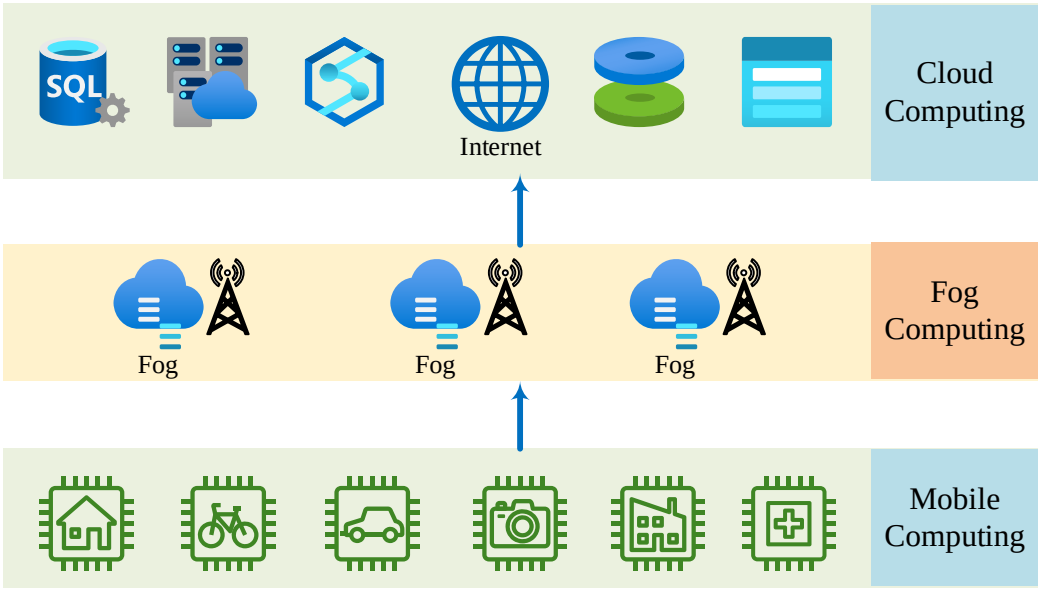

Figure 2 IoT Layer Stack

International Journal of Computer Networks and Applications (IJCNA)
DOI: 10.22247/ijcna/2023/220739 Volume 10, Issue 2, March – April (2023)

RESEARCH ARTICLE

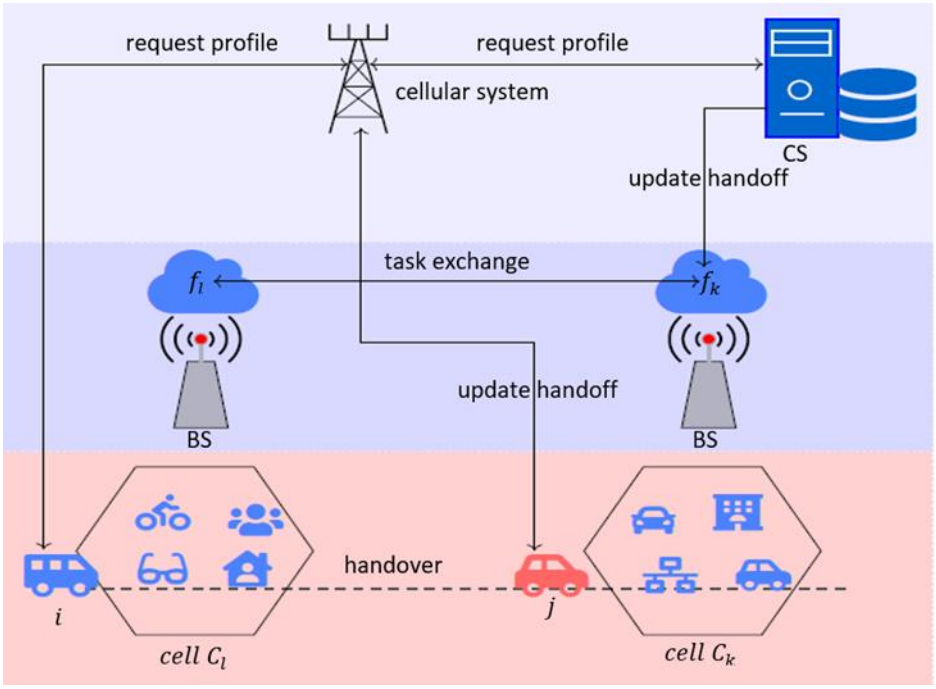

Figure 3 System Model

It is assumed that the CS has a set of Free Identifier List (FIL) from which the TDs can obtain their identifiers using getFreeId() function. The newly arriving TD i at a given c_k can know the idf of the fog f_k from other TDs residing in that cell.

The TDs generate tasks which are to be processed either locally at the TD or remotely at the fog nodes. When TD i in cell c_k offloads its task to fog f_k, the task response is sent out, after computation, from that fog f_k to TD i using either direct transmission, single hop, if the TD is still in the cell c_k or indirect transmission, multi-hop, through intervening fogs if the TD has moved (through handoffs) to another cell c_l, $l \neq k$.

4. FOG MODELING

It is assumed that TD arrivals at a given cell of a certain fog are of two types: type-1 or type-2. The first type comprises all TDs which join a given cell and are not associated with any fog (i.e., new arrival). Accordingly, TD i in Figure 1 is of type-1. The type-2 includes those TDs which are already associated with a certain fog but later moved (via handoff) to the coverage area of another fog. Accordingly, TD j in Figure 1 is of type-2. When type-1 arrives at a given cell, it registers with the CS which, in turn, creates a profile for it. To create a TD profile, the initialize massage iniMSG and replay message repMSG are used for coordination purposes between the TDs and the CS. When a type-2 TD moves from one cell to another via handoff, it updates its profile. For this updating, the update message updMSG is used. The task arrivals at fog f_k are distinguished into two classes: class-1 or class-2. Class-1 denotes arrivals from type-1 TDs. These task arrivals are queued at fog f_k in an infinite size buffer referred to as New Task Buffer (NTB).

To illustrate, consider the following scenario. TD i in cell c_k has a task request entry in the NTB buffer and is moving from the f_k coverage area toward cell c_l which is covered by the fog f_l, $k \neq l$. Using alert message alrMSG, the CS is in responsible for telling fog f_k that TD i has moved to cell c_l which is covered by fog f_l. In such case, the task of TD i which is queued in NTB buffer is re-classified as class-2 and moved into an infinite size buffer at fog f_k which is referred as Handoff Task Buffer (HTB). When the task of TD i is completed at fog f_k, the latter will send it to fog f_l which, in turn, will send it to TD i. In this paper, It is assumed that class-2 are given higher priority over, i.e. served before, class-1.

4.1. New TD Operations

When a TD i enters the transmission range of fog f_k, it senses the medium seeking beacon messages from other TDs residing in its cell c_k for a certain period of time τ. The beacon message bMSG contains the pair (idf, idc). Based on the sensing process result, TD i follows either one of the following two cases:

Case 1: The message bMSG is received by TD i within the time period τ. In such case, it records the pair (idf, idc) and initiates the message iniMSG = ⟨idf⟩ and sends it to the CS. Upon receiving iniMSG message, the CS, in turn, builds the

RESEARCH ARTICLE

profile $P_i^k = \langle idf, ifc, idt \rangle$, stors it in its DBS and replies to the new TD i with the message $repMSG = \langle idt \rangle$. The TD i combines the pair (idf, idc) with the received message $repMSG = \langle idt \rangle$ and consructs its profile $P_i^k = \langle idf, ifc, idt \rangle$.

Case 2: The time τ expires without receiving any beacon messages by TD i, meaning it is the first TD in cell c_k. As a result, it initiates the message $iniMSG = \langle x, y \rangle$ and sends it to the CS, where x and y are its coordinates in the cell c_k . When the CS receives the iniMSG message, it builds the profile $P_i^k = \langle idf, ifc, idt \rangle$, stores it in its data base and replies to TD i with the message $repMSG = \langle P_i^k \rangle$.

This procedure is illustrated in Algorithm 1.

Input: Time period τ

Output: Profile $P_i^k = \langle idf, ifc, idt \rangle$

TD i eneter cell k

TD i senses the medium for beacon messages

if A beacon message received within τ then

TD i sends the message $iniMSG = \langle idf \rangle$ to CS

The CS sets $idt = getFreeID()$

Construct profile $P_i^k = \langle idf, ifc, idt \rangle$

TD i receives its P_i^k via $repMSG = \langle P_i^k \rangle$

else

TD i sends $iniMSG = \langle x, y \rangle$ to CS

The CS sets $idt = getFreeID()$

TD i receives profile $repMSG = \langle idf, ifc, idt \rangle$

end

Algorithm 1 Profile Generation Process

Now it is time to handle offloaded tasks. Let $\mathcal{T}_i^k = (idt, T_P, T_D)$ be the offloaded task request at fog f_k, where T_P and T_D denote the task processing time and deadline time, respectively. Let $\mathcal{R}_i^k$ denote the response time of task $\mathcal{T}_i^k$. Clearly, the deadline T_D is an upper bound for $\mathcal{R}_i^k$. The offloaded task $\mathcal{T}_i^k$ at fog f_k is queued in the NTB buffer of the fog waiting for its processing turn. Let $\mathcal{V}_i^k$ be the cell dwell time of TD i, which is the time between the instant when TD i, residing in a cell c_k, offloaded its task to fog f_k and the instant when TD i is handed off to another cell c_l, $k \neq l$. Based on the values of $\mathcal{R}_i^k$, $\mathcal{V}_i^k$ and T_D, TD i receives its task response as follows:

1. If $\mathcal{R}_i^k \leq \mathcal{V}_i^k$ and $\mathcal{R}_i^k \leq T_D$, TD i will successfully receive its task response within its original cell c_k using direct transmission. In such case, the response time $\mathcal{R}_i^k$ is given as the sum of task uploading time U_i^k from TD i to fog f_k, queuing time Q_{NTB}^k in NTB buffer at fog f_k, the task processing time T_P at fog f_k and the task downloading time D_i^k from fog f_k to TD i. That is:

$$\mathcal{R}_i^k = U_i^k + Q_{NTB}^k + T_P + D_i^k \quad (1)$$

2. If $\mathcal{R}_i^k > \mathcal{V}_i^k$ and $\mathcal{R}_i^k < T_D$, TD i will not receive its task response within its original cell c_k. This scenario can occur as a result of TD i being handed off from cell c_k to another cell c_l. In this situation, one of the following two sub-scenarios applies:

- Scenario 1: TD i retransmits its task to its new fog f_l with new dealine. This retransmission will increase the response time, communication overhead and power consumption. In such scenario, the response time $\mathcal{R}_i^l$ is given as follows.

$$\mathcal{R}_i^l = U_i^l + Q_{NTB}^l + T_P + D_i^l + U_i^k + Q_{NTB}^k \quad (2)$$

- Scenario 2: To avoid the problem of increasing the response time, a transfer mechanism is used to migrate the task response from the old fog f_k to the new fog f_l in the case of handdoff process. In this mechanism, instead of retransmitting the task $\mathcal{T}_i^k$ to the new fog f_l, this task is completed at the old fog f_k and its response is transmitted to TD i via its new fog f_l using indirect transmission. In such scenario, the response time $\mathcal{R}_i^l$ is given as follows.

$$\mathcal{R}_i^l = U_i^k + T_P + Q_{HTB}^k + D_i^l \quad (3)$$

where Q_{HTB}^k is the queuing time in the HTB buffer at fog f_k. The transfer mechanism will be described in the next subsection.

4.2. Handoff TD Operations

Assume that TD i while in cell c_k and has offloaded its task $\mathcal{T}_i^k$ to fog f_k. While TD i waiting for task response from f_k, it starts moving on the highway toward the service coverage area of another fog f_l, $l \neq k$. When TD i receives strong wireless signal from f_l, it starts being handed off to f_l. After the automatic link transfer process is carried out, TD i begins to update its profile and starts the task migration mechanism as shown Figure 4.

First, TD i sends the update message $updMSG = \langle P_i^k \rangle$ to the CS. Then, the CS constructs the new profile P_i^l and replies to TD i with the message $repMSG = \langle P_i^l \rangle$. Note that the identifier idt of TD i is not changed form cell to cell. After TD i gets its profile at fog f_l, the CS starts the transfer mechanism as follows.

1) The CS sends an alert message $alrMSG = \langle P_i^k, P_i^l \rangle$ to the fog f_k containing the old and new profiles of TD i.

RESEARCH ARTICLE

2) The fog f_k searches its NTB buffer for the task $\mathcal{T}_i^k$ in profile is P_i^k.

3) If there is no an entry for P_i^k then fog f_k replies to fog f_l with message repMSG $= \langle P_i^l, \text{flag} = 1 \rangle$.

4) If there is an entry $\mathcal{T}_i^k$ for P_i^k then the fog f_k checks if that task can be processed within its deadline time T_D or not. This depends on the number of tasks queued in the HTB buffer.

5) If the task $\mathcal{T}_i^k$ can be processed within its deadline time T_D then fog f_k classifies it as class-2 and transfers it to the end of the HTB buffer.

6) Once the task $\mathcal{T}_i^k$ gets served, fog f_k replies to fog f_l with message repMSG $= \langle P_i^l, T_R, \text{flag} = 2 \rangle$, where T_R denotes the response of the task $\mathcal{T}_i^k$.

7) If the task $\mathcal{T}_i^k$ can not be processed within its deadline time T_D then fog f_k removes it from its NTB and replies to the fog f_l with message repMSG $= \langle P_i^l, \text{flag} = 3 \rangle$.

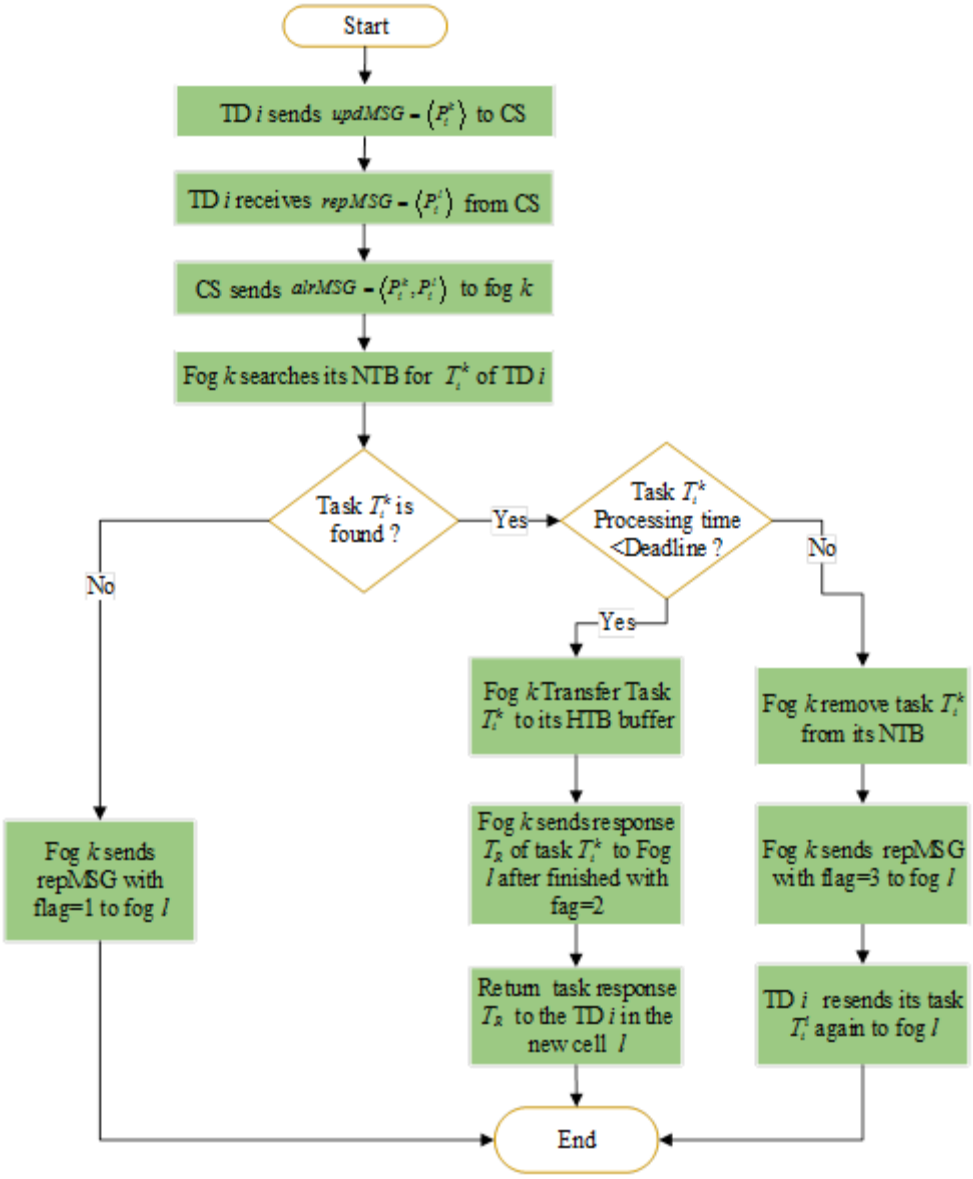

Figure 4 Handoff Process and Task Migration of TD i from cell k to Neighbor Cell l

International Journal of Computer Networks and Applications (IJCNA)
DOI: 10.22247/ijcna/2023/220739 Volume 10, Issue 2, March – April (2023)

RESEARCH ARTICLE

After receiving the repMSG message, fog f_l executes one of the following actions based on the value of flag:

- **Action 1,** if flag = 1**:** Do nothing.
- **Action 2,** if flag = 2**:** Fog f_l forwards the response T_R directly to TD i within the cell c_l.
- **Action 3,** if flag = 3**:** Fog f_l sends an alert to TD i to retransmit it task request again.

5. RESULTS AND DISCUSSIONS

In this section, the simulation and the results for the proposed collaborative task offloading framework are provided. The simulation program was coded in the Python language and later run on Intel(R) Core (TM) i5-10300H CPU @ 2.50GHz, with a Windows 10 pro version 21H2 operating system. Table 1 illustrates the different operational parameters that have been used to establish the simulation process. Simulation parameters are TDs task generation rate, ask processing time T_P at fogs, number of TDs within cells, number of VMs and handoff rate. In the simulation experiments, the handoff rate is used to control the speed of TDs descending on a fog. A low handoff rate means low mobility and a high handoff rate means high mobility. For a given cell k, the task uploading time U_k from TD to fog k and the task response downloading time D_k from the fog to TDs are fixed to 500 msec. Five million simulation runs were found sufficient for each experiment to achieve convergence. The simulation results are presented in terms of the task response time, as given by Equation (1), versus TD arrival rate, number of TDs in each cell, number of VMs and the task processing rate at fogs. In each simulation experiment, the task response time is measured in two cases, based on Equations (2) and (3): when the proposed framework is enabled and when it is disabled. In the event that a TD moves from one cell to another (i.e. handover), disabling the proposed framework prevents tasks from being migrated between fogs.

Table 1 Operational Parameters

Parameters 1	Value
Number of TDs	50-500
Number of VMs	50
Task upload time	500 msec.
Task response download time	500 msec.
Number of cells (fogs), N	3

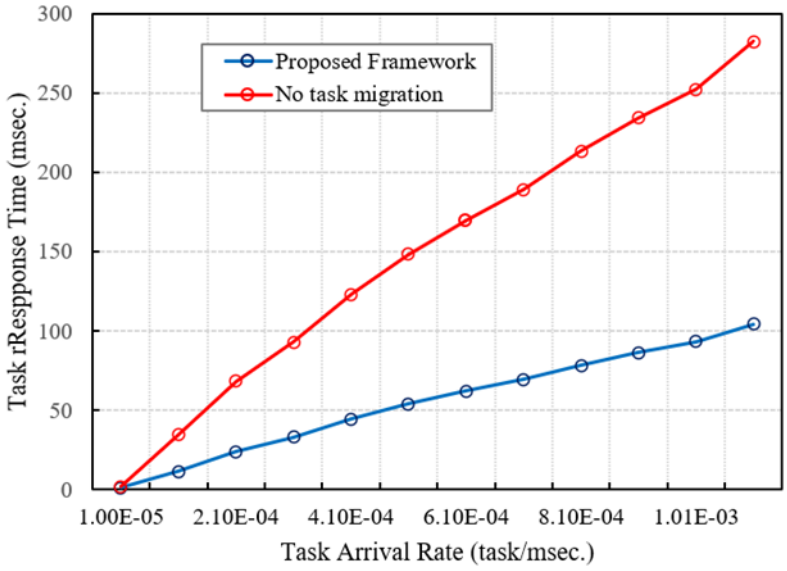

Figure 5 Task Response Time versus the Task Generation Rate in Low Mobility

Figure 5 shows how the proposed framework can improve the task response time in a high mobility situation as the handoff rate is 0.01 TDs/second. In this experiment, there are 500 TDs distributed uniformly over 3 cells. The number of VMs is 50 and the processing time rate at each VM is 0.001 task/second.

It can be seen that the task response time when using the framework (lower curve) is less than half its value without the framework (upper curve). The reason is that the framework allows a task that has been offloaded in some cell to continue being processed after its offloading TD departs the cell, whereas without the framework the TD would have to resubmit the task again to the new cell, dropping whatever processing done in the cell that has been departed. Another remarkable point about the framework is that the decrease it affords in response time becomes even more pronounced for higher task arrival rates.

RESEARCH ARTICLE

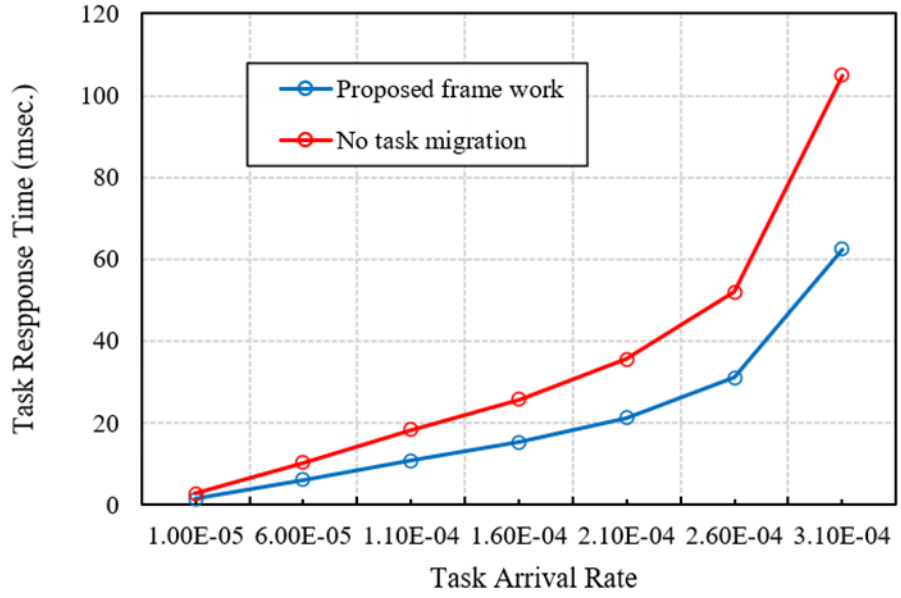

Figure 6 The Task Response Time versus the Task Generation Rate in Low Mobility

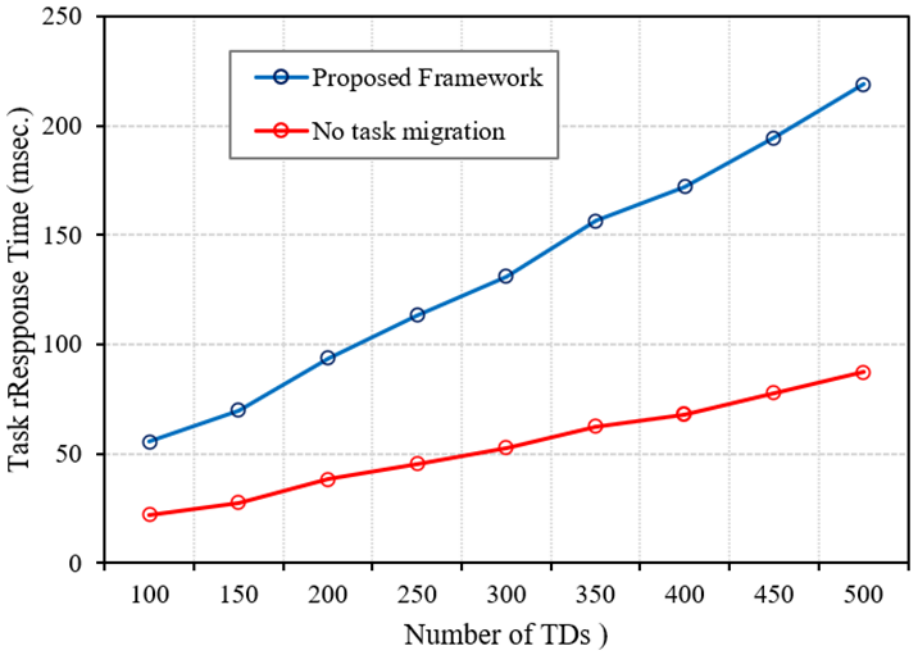

Figure 7 The Task Response Time versus the Number of TDs in High Mobility

Figure 6 is similar to Figure 5, but visualizes the case of a low mobility situation as the handoff rate is 0.00001 TD/second. The figure shows that the response time is less with the framework (lower curve) than without it (upper curve). However, by comparing Figures 5 and 6, it can be seen that the improvement introduced by the framework in the task response time is greater if the mobility is high than if it is low. The reason the proposed framework is more effective for high mobility than for low mobility can be interpreted as follows. When the mobility is low, the chances that the TD departs the cell where it has already offloaded a task is small, and hence the probability that it will receive the response before departing is high, obviating the need for the framework. On the contrary, when the mobility is high, the chances that the TD departs the cell where it has already offloaded a task is high, and hence the probability that it will not receive the

RESEARCH ARTICLE

response before departing is high, requiring the favor the framework provides.

Figure 7 illustrates the task response time against the number of TDs in high mobility, as the handoff rate is 0.01 TD/second, with and without the proposed framework. As can be seen, the framework helps decrease the task response time greatly regardless of how many TDs are offloading tasks. The reason, once again, is that migrating the task response to the next fog spares offloading task again at the new fog where the TD has moved.

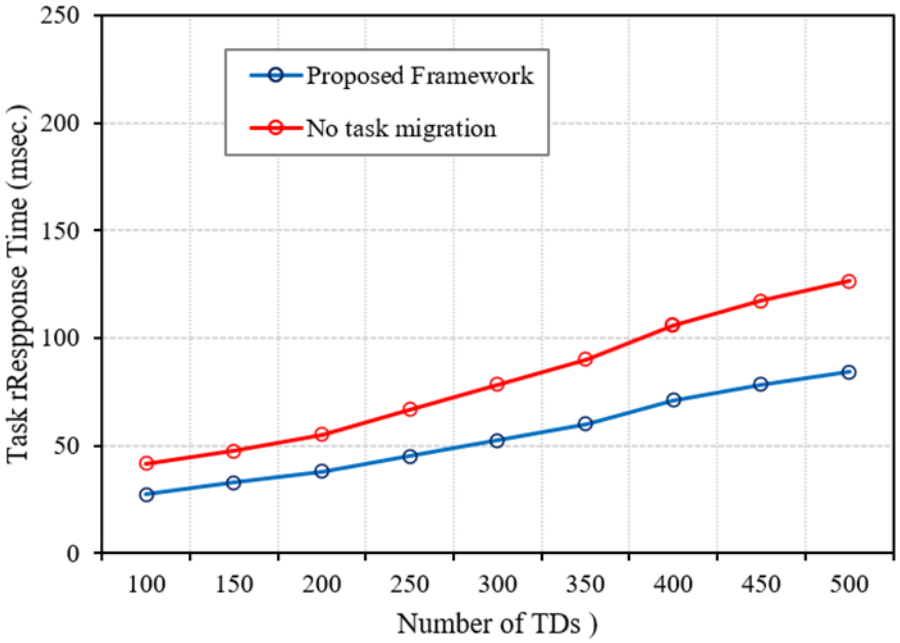

Figure 8 The Task Response Time versus the Number of TDs in Low Mobility

In low mobility, Figure 8, where the handoff rate is set at 0.00001 TD/second, the same favorable effect of the proposed framework can be seen, but this time to a lesser extent. The reason why the framework is less effective in low mobility is that, as mentioned before, the probability that a TD receives the response of a task that it has offloaded while the TD is still in the same offloading cell is high. This means that the service of the proposed framework is not as greatly needed as in the high mobility case.

6. CONCLUSION

In this article a collaborative cloud-fog framework is provided for task offloading of mobile TDs. The framework ensures seamless handover as the TD crosses boundaries from fog to fog, and also ensures low task response time satisfying stated QoS limits. The framework assumes a number of fog nodes deployed along a highway with a CS in the backbone network for monitoring the overall scene. The task response of the TD may migrate from fog to fog in case the TD is moving across various fogs. Two queues are provided in each fog, one to host tasks that are being locally offloaded and one to host tasks that have been handed over from other fogs. For the purpose of network integrity, an inter-fog messaging system to handle the work between fogs and the CS is introduced. Further, a mechanism for managing the handoff process of TDs in both high and low mobility is developed. A simulation program is written in Python to assess the performance of the proposed framework in terms of task response time, mobility, and cell dwell time. According to the simulation experiments carried out, the proposed framework lowers response time, and thus can help real-time applications meet stringent deadlines.

REFERENCES

[1] Cisco Annual Internet Report (2018–2023), [Online]. Available at : [https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.pdf]

[2] S. Smith, "IoT connections to grow 140% to hit 50 billion by 2022, as edge computing accelerates RoI," Juniper Research, 2018.

[3] D. Mendes et al., "VITASENIOR-MT: A distributed and scalable cloud-based telehealth solution," 2019 IEEE 5th World Forum on Internet of Things (WF-IoT), Limerick, Ireland, 2019, pp. 767-772, doi: 10.1109/WF-IoT.2019.8767184.

[4] K. Verma, A. Kumar, M. Salim Ul Islam, T. Kanwar, M. Bhushan, "Rank based mobility-aware scheduling in Fog computing", Informatics in Medicine Unlocked, vol. 24, 2021, 100619, doi.org/10.1016/j.imu.2021.100619.

RESEARCH ARTICLE

[5] Y. Kyung, "Performance Analysis of Task Offloading With Opportunistic Fog Nodes," in IEEE Access, vol. 10, pp. 4506-4512, 2022, doi: 10.1109/ACCESS.2022.3141199.
[6] J. Kuliga, S. Massicot, R. Adhikari, M. Ruppel, N. Jux, H.-P. Steinrück and H. Marbach, "Conformation Controls Mobility: 2H-Tetranaphthylporphyrins on Cu (111)," ChemPhysChem, vol. 21, p. 423–427, 2020, doi.org/10.1002/cphc.201901135
[7] 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, August 2012 , pp. 13–16, doi.org/10.1145/2342509.2342513
[8] M. Chiang and T. Zhang, "Fog and IoT: An Overview of Research Opportunities," in IEEE Internet of Things Journal, vol. 3, no. 6, pp. 854-864, Dec. 2016, doi: 10.1109/JIOT.2016.2584538.
[9] V. Dastjerdi, H. Gupta, R. N. Calheiros, S. K. Ghosh and R. Buyya, "Chapter 4 - Fog Computing: principles, architectures, and applications," in Internet of Things, R. Buyya and A. V. Dastjerdi, Eds., Morgan Kaufmann, 2016, pp. 61-75.
[10] P. Bellavista, J. Berrocal, A. Corradi, S. K. Das, L. Foschini and A. Zanni, "A survey on fog computing for the Internet of Things," Pervasive and Mobile Computing, vol. 52, pp. 71-99, 2019, doi.org/10.1016/B978-0-12-805395-9.00004-6.
[11] V. Dastjerdi and R. Buyya, "Fog Computing: Helping the Internet of Things Realize Its Potential," in Computer, vol. 49, no. 8, pp. 112-116, Aug. 2016, doi: 10.1109/MC.2016.245.
[12] F. Jalali, K. Hinton, R. Ayre, T. Alpcan and R. S. Tucker, "Fog Computing May Help to Save Energy in Cloud Computing," in IEEE Journal on Selected Areas in Communications, vol. 34, no. 5, pp. 1728-1739, May 2016, doi: 10.1109/JSAC.2016.2545559.
[13] N. Kumari, A. Yadav and P. K. Jana, "Task offloading in fog computing: A survey of algorithms and optimization techniques," Computer Networks, vol. 214, pp. 109137, 2022, doi.org/10.1016/j.comnet.2022.109137.
[14] R. Buyya,; S.Narayana Srirama, "Management and Orchestration of Network Slices in 5G, Fog, Edge, and Clouds," in Fog and Edge Computing: Principles and Paradigms , Wiley, 2019, pp.79-101, doi: 10.1002/9781119525080.ch4.
[15] Lakhan, M. Ahmad, M. Bilal, A. Jolfaei and R. M. Mehmood, "Mobility Aware Blockchain Enabled Offloading and Scheduling in Vehicular Fog Cloud Computing," in IEEE Transactions on Intelligent Transportation Systems, vol. 22, no. 7, pp. 4212-4223, July 2021, doi: 10.1109/TITS.2021.3056461.
[16] H. Raouf, R. Abdallah, H. Y. M. Soliman and R. Rizk, "Mobility-Aware Task Offloading Enhancement in Fog Computing Networks," in The 8th International Conference on Advanced Machine Learning and Technologies and Applications (AMLTA2022). AMLTA 2022, vol 113. Springer, Cham, doi.org/10.1007/978-3-031-03918-8_47.
[17] F. Chiti, R. Fantacci and B. Picano, "A Matching Theory Framework for Tasks Offloading in Fog Computing for IoT Systems," in IEEE Internet of Things Journal, vol. 5, no. 6, pp. 5089-5096, Dec. 2018, doi: 10.1109/JIOT.2018.2871251.
[18] Z. Zhao et al., "On the Design of Computation Offloading in Fog Radio Access Networks," in IEEE Transactions on Vehicular Technology, vol. 68, no. 7, pp. 7136-7149, July 2019, doi: 10.1109/TVT.2019.2919915.
[19] 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," in IEEE Transactions on Communications, vol. 66, no. 4, pp. 1594-1608, April 2018, doi: 10.1109/TCOMM.2017.2787700.
[20] S. Misra and N. Saha, "Detour: Dynamic Task Offloading in Software-Defined Fog for IoT Applications," in IEEE Journal on Selected Areas in Communications, vol. 37, no. 5, pp. 1159-1166, May 2019, doi: 10.1109/JSAC.2019.2906793.
[21] L. Liu, Z. Chang, X. Guo, S. Mao and T. Ristaniemi, "Multiobjective Optimization for Computation Offloading in Fog Computing," in IEEE Internet of Things Journal, vol. 5, no. 1, pp. 283-294, Feb. 2018, doi: 10.1109/JIOT.2017.2780236.
[22] J. Yao and N. Ansari, "QoS-Aware Fog Resource Provisioning and Mobile Device Power Control in IoT Networks," in IEEE Transactions on Network and Service Management, vol. 16, no. 1, pp. 167-175, March 2019, doi: 10.1109/TNSM.2018.2888481.
[23] Yousefpour, G. Ishigaki, R. Gour and J. P. Jue, "On Reducing IoT Service Delay via Fog Offloading," in IEEE Internet of Things Journal, vol. 5, no. 2, pp. 998-1010, April 2018, doi: 10.1109/JIOT.2017.2788802.
[24] S. Jošilo and G. Dán, "Decentralized Algorithm for Randomized Task Allocation in Fog Computing Systems," in IEEE/ACM Transactions on Networking, vol. 27, no. 1, pp. 85-97, Feb. 2019, doi: 10.1109/TNET.2018.2880874.
[25] S. Ghosh, A. Mukherjee, S. K. Ghosh and R. Buyya, "Mobi-IoST: Mobility-Aware Cloud-Fog-Edge-IoT Collaborative Framework for Time-Critical Applications," in IEEE Transactions on Network Science and Engineering, vol. 7, no. 4, pp. 2271-2285, 1 Oct.-Dec. 2020, doi: 10.1109/TNSE.2019.2941754.
[26] M. M. Razaq, B. Tak, L. Peng and M. Guizani, "Privacy-Aware Collaborative Task Offloading in Fog Computing," in IEEE Transactions on Computational Social Systems, vol. 9, no. 1, pp. 88-96, Feb. 2022, doi: 10.1109/TCSS.2020.3047382.
[27] M. Keshavarznejad, M. H. Rezvani and S. Adabi, "Delay-aware optimization of energy consumption for task offloading in fog environments using metaheuristic algorithms," Cluster Comput vol. 24, pp.1825–1853 (2021). https://doi.org/10.1007/s10586-020-03230-y.
[28] Qayyum, T., Trabelsi, Z., Waqar Malik, A. et al. Mobility-aware hierarchical fog computing framework for Industrial Internet of Things (IIoT). Journal of Cloud Computing vol. 11, no. 72 (2022). https://doi.org/10.1186/s13677-022-00345-y
[29] Rahbari, D., Nickray, M. Task offloading in mobile fog computing by classification and regression tree. Peer-to-Peer Networking and Applications, vol. 13 , pp. 104–122 (2020). https://doi.org/10.1007/s12083-019-00721-7
[30] Z. Ning et al., "Partial Computation Offloading and Adaptive Task Scheduling for 5G-Enabled Vehicular Networks," in IEEE Transactions on Mobile Computing, vol. 21, no. 4, pp. 1319-1333, 1 April 2022, doi: 10.1109/TMC.2020.3025116.
[31] ALVI, Ahmad Naseem, et al. Intelligent task offloading in fog computing based vehicular networks. Applied Sciences, 2022, vol. 12, no. 9, 4521, https://doi.org/10.3390/app12094521
[32] Hosseini, E., Nickray, M. & Ghanbari, S. Energy-efficient scheduling based on task prioritization in mobile fog computing. Computing vol. 105, pp. 187–215 (2023). https://doi.org/10.1007/s00607-022-01108-y
[33] Aisha Muhammad A. Hamdi, Farookh Khadeer Hussain, Omar K. Hussain, Task offloading in vehicular fog computing: State-of-the-art and open issues, Future Generation Computer Systems, vol. 133, 2022, pp. 201-212, https://doi.org/10.1016/j.future.2022.03.019.

Authors

Amira S.Ibrahim received the B.Sc. degree in Computer Science from the Faculty of Computer and Informatics, Suez Canal University. Currently, she pursuing the Ph.D. degree with the Department of the Computer Science, Suez Canal University. Her research interests include cloud computing, computer security and computer network.

Hassan AL-Mahdi received the BSc in Computing Science, the MSc in Computer Science and Ph. D. in Wireless Networks from the Faculty of Science, Suez Canal University, Egypt in 1994, 2001 and 2005, respectively. He is a Full Professor of Computer Networks at the Faculty of Computer and Informatics, Suez Canal University, Egypt. His researches are in fields of ad hoc networks, mobile cellular communications, cognitive radio networks, IoT, Cloud Computing, WSN, and the performance evaluation of computer networks. He has many international papers mostly in the area of performance evaluation of computer networks, IoT, WSN, Cloud Computing, Queuing System and Cryptography. He can be contacted at email: drhassanwesf@ci.suez.edu.eg.

RESEARCH ARTICLE

Hamed Nassar received the B.Sc. degree in electrical engineering from Ain Shams University, Egypt, in May 1979, and the M.Sc. degree in electrical engineering and the Ph.D. degree in computer engineering from the New Jersey Institute of Technology, USA, in May 1985 and May 1989, respectively. He has been a full professor in the Department of Computer Science, Suez Canal University, Egypt, since 2004. Besides Egypt, he has taught computer science and engineering courses in USA, Lebanon and Saudi Arabia. Dr. Nassar has published numerous articles in international journals and conferences. His research interests include mathematical modelling of computer and communications systems, cloud computing and machine learning.

How to cite this article:

Amira S. Ibrahim, Hassan Al-Mahdi, Hamed Nassar, "A Collaborative Offloading Task Framework for IoT Fog Computing", International Journal of Computer Networks and Applications (IJCNA), 10(2), PP: 244-255, 2023, DOI: 10.22247/ijcna/2023/220739.