Mobility Support for Energy and QoS aware IoT Services Placement in the Fog

Tanissia Djemai, Patricia Stolf, Thierry Monteil, Jean-Marc Pierson

Mobility Support for Energy and QoS aware IoT Services Placement in the Fog

Tanissia Djemai
IRIT, LAAS-CNRS
University of Toulouse
Toulouse, France
tanissia.djemai@irit.fr

Patricia Stolf
IRIT
University of Toulouse
Toulouse, France
patricia.stolf@irit.fr

Thierry Monteil
LAAS-CNRS
University of Toulouse, CNRS, INSA
Toulouse, France
monteil@laas.fr

Jean-Marc Pierson
IRIT
University of Toulouse
Toulouse, France
jean-marc.pierson@irit.fr

***Abstract*—Fog computing has emerged as a strong distributed computation paradigm to support applications with stringent latency requirements. It offers almost ubiquitous computation capacities over a large geographical area. However, Fog systems are highly heterogeneous and dynamic which makes services placement decision quite challenging considering nodes mobility that may decrease the placement decision quality over time. This paper proposes a Mobility-aware Genetic Algorithm (MGA) for services placement in the Fog which aims at supporting nodes' mobility while ensuring both infrastructures energy-efficiency and applications Quality of Service (QoS) requirements. We have compared this approach with two variants of Shortest Access Point migration strategy (SAP) from the literature, a proposed Mobility Greedy Heuristic (MGH) and a baseline Simple Genetic Algorithm (SGA). Experiments conducted with MyiFogSim simulator have shown that MGA ensures good performances in terms of energy and delay violations minimization compared to other methods.**

***Index Terms*—Internet of Things, Optimization, Mobility, Fog Computing, Smart Campus, QoS, Energy.**

I. Introduction

Fog computing is a distributed computation paradigm that extends cloud infrastructures with computation and storage capabilities of nodes located between end users and data centers [1]. It supports latency-constrained applications, by hosting services and data as close as possible to end-users and leads to cloud-network traffic reduction . In the context of Internet of Things (IoT), the Fog infrastructure, composed by constrained computation and communication nodes, has to deal with a significant number of mobile IoT objects. Those objects may request at any time applications with heterogeneous computation and communication needs and context-usage priorities. Placing those applications considering the system dynamic induced by nodes mobility is quite challenging and needs to be addressed to ensure and maintain the Quality of Service (QoS) of applications over time. Moreover, the distributed and large-scale characteristics of IoT and Fog environments lead to a non negligible impact on energy consumption [4] that should be considered as a prime parameter for any Fog-IoT services placement framework.

Mobility support has mainly been investigated with handover/handoff mechanisms and migration process, [11], [12], defined respectively as the process of managing the connectivity of users between network cells and virtual services transfer from one computing node to another. Managing the mobility of nodes with migration induces many questions such as: "When the migration should start?", "Where to place services?" and "How to choose the data transfer path?". The migration process undeniably induces an extra processing time, network usage and energy consumption. Furthermore, if the requested IoT applications are designed in a distributed manner and hosted by different data-dependent virtual entities, it will be necessary to choose which and how services should be migrated to ensure the communication between them during the migration. In order to avoid the migration process and its previous cited drawbacks, this paper proposes a Mobility-aware Genetic Algorithm (MGA) and a Mobility Greedy Heuristic (MGH) as placement methods that consider mobility patterns of nodes while minimizing the infrastructure energy consumption and ensuring the QoS of applications. The contributions of this work can be summarized as follows: **(1)** A probabilistic Genetic Algorithm and a greedy heuristic approaches for IoT-Fog services placement with nodes mobility support. **(2)** Performance comparison between different services placement approaches. The metrics considered are energy consumption and applications delay violations. **(3)** An extension of MyiFogSim simulator that integrates some literature mobility patterns. The remainder of this paper is structured as follows : In Section II the state of the art of Mobility and Fog computing services placement is addressed. Section III gives the motivating scenario and a formalization of the placement problem. Section IV details the proposed placement methods. Finally, Section V reports experimental results using MyiFogSim.

II. Related Work

Fog-Services placement problem has been widely investigated within a static Fog context (without mobility).

Fog and services placement problem in a static environment: Services placement problem in the Fog is commonly defined as an optimization problem to ensure one or multiple system QoS requirements such as response time, resource usage efficiency, network consumption minimization. For this purpose, different approaches have been used such as evolutionary optimization, various heuristics and exact resolutions under different system constraints [2], [3], [20], [21], [22], [23]. Those works focus only on QoS requirements in a static Fog environment.

Fog and Energy: Authors in [5] try to minimize mobile nodes energy consumption, time and execution costs by transferring their tasks to complementary computation devices. This approach can induce to an excessive usage of bandwidth and increases the energy consumption of other nodes. Work in [6] addresses the IoT to Fog nodes assignment problem with Genetic Algorithm (GA) and Broadcast Incremental Power (PIP) algorithm to reduce mobile nodes energy consumption under delay constraints. The approach does not consider the energy consumption of Fog, cloud nodes and the network. In [8], authors use a Discrete Particle Swarm Optimization

approach to minimize the infrastructure energy consumption and applications delay violations in a static Fog environment.

Fog and Mobility support: Although mobility is a key point to consider in the Fog, only few works investigate it. Authors in [7] consider the influence of nodes mobility for applications scheduling problem; however, the mobility information is not used by the algorithms and the mobility scenario is only based on increasing and decreasing the number of connected nodes and does not consider any mobility model. Authors in [9] propose an Integer Linear Program to minimize applications makespan while considering a random way point mobility model. This model neglects the temporal and spatial aspects that define most of IoT nodes motion. In [10], authors propose to dynamically re-schedule the placement of Virtual Network functions (VNF) to minimize end-to-end latency of users and the number of migrations. Compared to our work, this approach is only based on latency violation metric. None of these works introduces the mobility information in the placement modeling and algorithm.

III. System Description and Problem Statement

This section gives details about the considered scenario, the system model and the problem formulation.

A. Motivating Scenario, Assumptions and Mobility Model

We consider a Smart Campus area with pedestrians carrying IoT nodes that request applications and interact with their sensors and actuators. For instance, a smartphone can request for an augmented reality (AR) application. The processing services of the application will interact with the camera of the smartphone as a source sensor and consider the smartphone screen as the actuator. We aim to place the AR services considering the mobility of pedestrians.

For sake of simplicity, but without loss of generality, we make the following assumptions: **(1)** We consider a time-slotted system $\mathbb{T} = \{t_0, t_1, .., t_{T-1}\}$. One slot duration is defined as half the time taken by the fastest node in the system to cross the shortest distance between two points and that one node movement is negligible within one slot i.e. one node can only change position at the beginning of a time t_i. **(2)** We suppose that the service duration is infinite and that if two IoT user nodes [1] u_1 and u_2 request the same application a_1, each node will have its own virtual instances (i.e instances are not shared between two IoT nodes). **(3)** Following 5G specifications, we assume that a user u will always be under network coverage and it will be attached to one and only one access point at each time $t \in \mathbb{T}$. **(4)** We assume that communications between IoT users nodes and their services are uniformly distributed within the time window $W_{\mathbb{T}}$, that represents the studied time interval of T slots.

In environment that involves humans, mobility patterns are not as uncertain as in other systems. Moreover, with access technologies such as 5G and existing 4G-LTE , we may not need a precise location of nodes to ensure good services and mobility management. In our approach, we use the Weighted Waypoints mobility model (WWP) that considers location, pause duration and weights to choose the next destination place. It has been built from a mobility survey carried out on the campus of the University of Southern California. The WWP defines a set of locations called waypoints. The probability that mobile nodes move from waypoint a to waypoint b depends on current location and time [16].

[1]"IoT user node" or "user node" terms refer to a node that has requested an application and "node" term refers to any computation node that includes also the IoT nodes.

B. System Representation

1) Physical topology: We consider a hierarchical Fog infrastructure described by an non directed graph $\mathbb{G} = (\mathbb{M}, \mathbb{L})$ and shown in Figure 1, where $\mathbb{M}$ is the set of nodes, and $\mathbb{L}$ the set of direct links between them.

Each physical node $m_k \in \mathbb{M}$ is defined by: *(1)* a type $\eta_k \in \{IoT, Fog, Cloud\}$ *(2)* a capacity vector $\Omega_k =< cpu_k^{max}, ram_k^{max}, disk_k^{max} >$, parameters units of elements are respectively in MIPS, MB and MB. *(3)* a 2D coordinates and a coverage zone vector at time t, $\Delta_k^t =< x_k^t, y_k^t, r_k^t >$.

Nodes have additionally characteristics depending on their type: *(1) IoT node*: It has computation and storage capacities and a set of sensors $\mathbb{H}_k^0$ and actuators $\mathbb{H}_k^1$ components. IoT nodes move according to a probability mobility model ϑ_k (e.g. Manhattan model) [16], and have a set of requested applications $\mathbb{A}_k^t$ at time $t \in \mathbb{T}$. The coverage zone is fixed to $r_k^t = 0$. Let $\mathbb{U}$ be the subset of $\mathbb{M}$ with only IoT nodes. *(2) Fog node*: It has the ability to connect IoT nodes to external network through Access point equipment and has computation and storage capacity through a server cloudlet device. *(3) Cloud node*: It is defined as a limitless computation data center server. The coverage zone is fixed to $r_k^t = 0$.

Each network link $l_{(k,k')} \in \mathcal{L}$ between nodes m_k and $m_{k'}$ is defined with its bandwidth $bw_{(k,k')}$ in (Mb/s) and latency $lc_{(k,k')}$ in (ms). $l_{(k,k')}$ can be one direct link from m_k to $m_{k'}$ or be composed by several links.

2) Internet of Things applications: We consider a set of IoT applications $\mathbb{A} = \{a_0, .., a_i, .., a_{A-1}\}$ designed in a distributed manner. Each application $a_i \in \mathbb{A}$ has a class type Υ which can be: Mission Critical (MC), Real Time (RT), Streaming (ST) and Best Effort (BE), having respectively priority level ψ of 0, 1, 2, and 3 and response delay requirement ϕ of 20ms, 50ms, 150ms and ∞, details are given in [8].

An application a_i has a set of size n_i of communicating data-dependent services represented with a directed graph $\mathbb{G}(a_i) = (\mathbb{S}^{a_i}, \mathbb{E}^{a_i})$. Each service $s_j \in \mathbb{S}^{a_i}$ is defined by its requested CPU mi_j (MIPS) and RAM ram_j (MB), deployment technology tec_j (either a virtual machine, a container, OSGi plugin, Java Virtual Machine or a combination of them). We consider two specific nodes in the graph $\mathbb{G}(a_i)$ that represent respectively sensor sending service (source node) and actuator receiving service (sink node).

Each directed edge $e_{(j,j')} \in \mathbb{E}^{a_i}$, from service s_j to service $s_{j'}$ represents a data dependency between s_j and $s_{j'}$ and carries $data_{(j,j')}$, the volume of data sent from s_j to $s_{j'}$ in (KB). We would like to draw the reader's attention to the fact that we have willingly ignored the application index i while using the notation s_j rather than s_{ij} to lighten the model.

C. Problem Formulation and Objective Function

We consider a set of mobile IoT nodes $\mathbb{U}$ requesting a set of applications $\mathbb{A}^{t_0}$ at initial time t_0. Each IoT node $u \in \mathbb{U}$ has a set $\mathbb{A}_u^{t_0}$ of requested applications and $\mathbb{N}_u^{t_0}$, the set of all services in $\mathbb{A}_u^{t_0}$. Let's consider the binary decision variable x_{jk}^u equals to 1 if service $j \in \mathbb{N}_u^{t_0}$ requested by node u is placed on node k and equals to 0 otherwise.

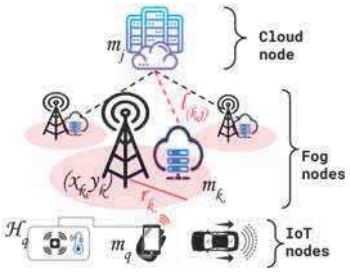


Fig. 1. Physical Topology representation highlighting the system's variables

1) Energy consumption: The first metric we consider, defined in Eq (1), is the sum of computation and network energy consumption during T time slots. As the energy model does not impact our resolution approach, we have chosen, similarly to work [17] a linear model, but other models could be used [19].

$$f_1 = f_C + f_N \tag{1}$$

The computation part is estimated according to the following equation:

$$f_C = T \sum_{u \in \mathbb{U}} \sum_{j \in \mathbb{N}_u} \sum_{k \in \mathbb{M}} x^u_{jk} \alpha^u_{jk} \gamma^c_k \tag{2}$$

α^u_{jk} is the computation time of service instance j associated to the IoT user node u and placed on node k as defined in Eq (3). As the placement is made at the beginning and will remain unchanged, the computation cost is time independent. Placing service j on node k depends on the maximum processing capacity of the physical node cpu^{max}_k in MIPS, the amount of MIPS allocated to other services placed on the machine k and the amount of requested MIPS of service j.

$$\alpha^u_{jk} = \frac{mi_j + \sum_{u \in \mathbb{U}} \sum_{l \in \mathbb{N}^u - \{j\}} mi_l x^u_{lk}}{cpu^{max}_k} \tag{3}$$

γ^c_k is the computation power of node k: It represents the difference between node maximum and minimum power consumption $\gamma^c_k = Pc^{max}_k - Pc^{min}_k$. The communication part is estimated according to the following equation:

$$f_N = \sum_{z \in \mathbb{M}-u, t \in \mathbb{T}} P^t_{uz} \sum_{j,j' \in \mathbb{N}_u} \sum_{k,k' \in \mathbb{M}} x^u_{jk} x^u_{j'k'} \beta^{ut}_{jj',kk'} \gamma^n_{kk'} \tag{4}$$

$\beta^{ut}_{jj',kk'}$ defines the communication time between service instances j and j' placed respectively on nodes k and k' at time t for user node u. In contrast to the computation time, the communication time $\beta^{ut}_{jj',kk'}$ varies over time, because it is impacted by the fluctuation of latency between the mobile node and its processing services placed on the infrastructure. There are two possible causes for this fluctuation : *(1)* The IoT node does not host services; then the latency variation is due to the communication between sensors and their destination services and actuators and their source services. *(2)* The IoT node hosts services; then the communication latency between those services and other services will vary over time depending on the localization of the IoT node's.

$$\beta^{ut}_{jj',kk'} = \frac{1}{T} \frac{data_{(j,j')}}{bw^t_{(k,k')}} + lc^t_{(k,k')} \tag{5}$$

$\gamma^n_{kk'}$ is the communication power constant between node k and node k' and is defined by $\gamma^n_{kk'} = Pn^{max}_k + Pn^{max}_{k'} - Pn^{min}_k - Pn^{min}_{k'}$, where Pn^{max} and Pn^{min} are respectively the maximum and minimum power consumption of nodes network interfaces.

P^t_{uz} is the probability that an IoT node u is under the coverage zone of a Fog node z at time t. Its computation varies according to the chosen mobility model. This work considers the Weighted Waypoints Model (WWP). Considering a set of Waypoints and Access points located in a 2D area, we compute the Euclidean distance between each waypoint and Access point. Then, under the assumption that each waypoint is attached to one and only one Access point, we associate Waypoints to their closest Access point. P^t_{ur} is the probability that user u will be in waypoint r at time t and is equal to P_{uz} previously defined. $P^t_{uz} = P^t_{ur}$.

2) Delay violations : The second considered metric is the QoS violation metric defined in Eq (6) by the average number of applications delay violations.

$$f_2 = \frac{1}{T} \sum_{u \in \mathbb{U}} \sum_{a \in \mathbb{A}} w^t_a(u) \tag{6}$$

where $w^t_a(u)$ is equal to 1 if the application response delay $d^t_a(u)$ is greater than its maximum delay requirement D_a. Otherwise it is equal to 0. The maximum response delay D_a of the application a varies according to its class type Υ as presented in III-B. The estimated response delay $d^t_a(u)$ corresponds to the time between the sending of one data packet from the sensor of node u and its corresponding action on the actuator. The delay depends on the position of the IoT nodes and the placement of the used services. For each instance of the application $a \in \mathbb{A}$ requested by an IoT user node $u \in \mathbb{U}$, we compute the $d^t_a(u)$ according to the possible position of the user at time t as follows:

$$\begin{aligned} d^t_a(u) = & \sum_{j \in \mathbb{N}_u} \sum_{k \in \mathbb{M}} x^u_{jk} \alpha^u_{jk} \\ & + \sum_{z \in \mathbb{M}-\{u\}} P^t_{uz} \sum_{j,j' \in \mathbb{N}} \sum_{k,k' \in \mathbb{M}} x^u_{jk} x^u_{j'k'} \beta_{jj',kk'} \end{aligned} \tag{7}$$

3) Objective function: We combine considered the objectives into a single function, defined in Eq (8), to have Mono-objective (MAM) optimization approach which is recommended in many works such as [18], instead of computing the Pareto front that could be time consuming. We aim to find a solution that is a trade-off between the two objectives and gives a little advantage to the energy aspect.

$$F = f_1(1 + f_2) \tag{8}$$

IV. Placement Algorithms

A. Mobility-aware Genetic Algorithm (MGA)

Genetic Algorithm (GA) is an evolutionary based optimization approach, efficiently used for NP-hard problems resolution [13]. It has shown its efficiency in static Fog environments placement problems [20]. We propose to apply the GA method with a probabilistic objective function enhanced by nodes mobility information. A chromosome c, is a vector representing the placement solution. Each vector element is indexed by the service Id s^u_j and gives the node Id m_k that will host the service. IoT users nodes are sorted by the increasing

number of requested applications. The applications of each user node are sorted by the increasing priority values, and services of each application are sorted by increasing CPU demand. Through experiments we have chosen the following parameters values: mutation rate $\lambda = 0.01$, crossover rate $\sigma = 0.8$, population size $P = 20$, number of generations $\mu = 20$ and a 1-way tournament selection. The fitness of chromosomes is computed according to Eq (8) where the mobility model is used. For each user node u that requested an application at time slot t_0 and for each time step $t \in \mathbb{T}$ of the time window, we get the mobility information of u. The mobility information P_{uk}^t is the probability that the user node u is under the coverage zone of Fog node k. MGA is presented in Pseudo Algorithm (1).

Algorithm 1 Mobility-aware Genetic Algorithm (MGA) for IoT service placement

Get mobility information for each mobile node that has requested an application at time t_0.
Generate uniformly the initial population of size P and compute the fitness value for each chromosome with Eq (8).
while The number of generations μ is not reached **do**
  while Next generation population size does not reach σP individuals **do**
    On $(100 * \sigma)\%$ of the population apply 1-way tournament selection to select a pair of parents chromosomes c_1, c_2.
    Apply Crossover operation on c_1, c_2 to get a resulting new chromosome c_3 by taking first genes from c_1 and the rest from c_2.
    Apply mutation operation on c_3 with probability λ.
    Compute fitness of c_3 according to Eq (8) where the probabilistic part related to mobility is detailed in Eq (4).
    Put c_3 in the next generation population
  end while
  Keep $(100 * (1 - \sigma))\%$ of the best individuals from the previous population and add them to the new generation population.
  Increment the number of generations (μ value).
end while=0

B. Mobility Greedy Heuristic (MGH)

The main idea of the MGH heuristic is to consider the most probable path for each mobile device and then compute the placement that will reduce the energy consumption and communication delay based on those paths. MGH steps are as follows: *(1)* The most probable path is composed greedily by taking the next position that has the highest probability value within the transition matrix of the WWP model. *(2)* User nodes are sorted by increasing number of applications demand and then services within each application are sorted first by data dependency (within sensor to actuator data path), and second by increasing CPU demand. *(3)* For each time $t \in T$, user $u \in \mathbb{U}$, application $a_i \in \mathbb{A}_u$, we try to place $s_j \in a_i$. We estimate system computation energy consumption f_C, communication time d_a^t and communication energy consumption f_N of the sub-graph composed of previous placed services and the service s_j. The computed values of f_C ,f_N and d_a^t are normalized and then the chosen placement is the one that gives the minimum value of $G = \frac{1}{T}(f_C + f_N + d_a^t)$. The placement of s_j is never reconsidered when placing the remaining services. MGH has a different fitness formula from MGA one but it has same objectives : minimize delay violations and energy consumption. MGA approach places services of all applications at the same time, so it is possible to estimate delay violations in the set of applications. In contrast, the MGH places services gradually and for each new service to place it tries to minimize greedily the communication delays between this last and previous placed services.

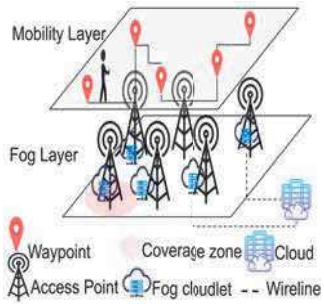


Fig. 2. Experiments map representing mobility and the Fog layers

(a) (b)

Fig. 3. Applications topologies : (a) Sequential Data flow and (b) Master-Slave Data flow processing graphs.

V. Evaluation

A. Experimentation description

For the validation of our placement approaches, we propose a Fog infrastructure deployed in a $18Km^2$ square geographical area, represented in Figure 2 and simulated with MyiFogSim [15]. Experiments have been conducted on an Intel Core i5-7200U CPU @ 2.50GHz x 4. MyiFogSim has been extended to allow the integration of various mobility models. As it is shown in Figure 2 from section III-A, the computing infrastructure is composed of fixed-located Fog nodes at each waypoint in the mobility layer. Fog nodes are connected to the cloud with wireline links and they are reachable by IoT nodes via access point devices. Nodes capacities values are taken uniformly within intervals specified in Table I. Each IoT node (smartphone) requests an application chosen uniformly within applications class set defined in Section III-B2. Each application a_i has a services set size of $n_i = 3$ and its graph is either modeled with a Sequential Unidirectional or a Master Slave data flow processing graphs with one sensor as data source and one actuator as sink. The graphs are represented in Figure 3. Service demand in terms of processing, memory and storage are taken from applications presented in [14]. The time window $W_{\mathbb{T}}$ is composed of T homogeneous time slots. At t_0, IoT nodes request their applications (one application per IoT node). For the mobility aspect, we consider several pedestrians carrying smartphones moving between five waypoints in the campus according to the Weighted Waypoints model. The considered waypoints are: "Classroom", "Library", "Cafeteria", "Other area on the campus", "Off campus area". Pause time distributions and transition for each waypoint and transition matrices are detailed in [16]. At t_0, IoT mobile nodes are uniformly distributed between the waypoints. We compare the proposed placement methods with a Simple Genetic Algorithm (SGA) and with two other strategies using migration from the literature (SAP and K-SAP). *(1) Static Genetic Algorithm* (SGA) has the same parameters values and fitness function as MGA but it does not consider the mobility model of IoT nodes. That means that for each IoT node $u \in \mathbb{U}$, Fog node $k \in \mathbb{M}$ and time $t \in \mathbb{T}$ we have $P_{uk}^t = P_{uk}^{t_0}$. *(2) Shortest Access Point migration Server Cloudlet* (SAP) consists in placing initially the requested services on the Fog node which is associated to the IoT node

closest Access point. Then at each time slot, those services are migrated to the distance-closest cloudlet (Fog node) following user's nodes path through Access points. *(3) K-Users Shortest Access Point migration Server* (K-SAP) consists of applying SAP migration process for K mobile nodes chosen randomly. For the other nodes, services will remain placed on Fog nodes associated to their closest Access point at t_0. The minimum value of parameter K is 0 and gives us a non migration strategy (a static placement based on initial position of nodes). The maximum value of K is the maximum number of mobile nodes and leads the K-SAP to act as the SAP strategy. After conducting experiments with various values of K we have chosen to present the most advantageous configuration for K-SAP which is K equal to 50% of IoT users nodes. We set the time window $W_{\mathbb{T}}$ to $1000 * 60 * 30$ with a time slot duration of $1000 * 60 * 5$ respectively equivalent to 30 minutes and 5 minutes. We vary uniformly the characteristics of nodes according to values in Table I 10 times. For each infrastructure configuration we vary IoT nodes size from 2 to 100. For each infrastructure configuration and IoT nodes size, we vary mobility seeds 30 times. For each infrastructure configuration, IoT nodes size and mobility seed, we vary the characteristics of applications 30 times with priority values and maximum response delay taken uniformly within values given in Section III-C, graphs are taken uniformly between Maser-Slave and Sequential unidirectional processing data flow. Computation and storage demands of services are taken from [14]. Presented results are the average values by IoT nodes size experiment. We would like to draw the reader's attention to the fact that those experiments focus on the mobility aspect of IoT nodes and the placement of their associated services. For this reason, the dimension of the problem that is increased is the number of IoT nodes and consequently the number of requested application.

TABLE. I.
INFRASTRUCTURE NODES DETAILS

Node	Number	CPU Interval (MIPS)	RAM Interval (MB)	Min-Max CPower[a]	Min-Max NPower[b]	Up Bandwidth Interval(GB/s)
Cloud	1	$[2400000, 9600000]$	$[25000, 100000]$	[320.339-120.43]	[60.21-160.16]	-
Fog	6	$[1400000, 5600000]$	$[12500, 50000]$	[107.339-83.433]	[41.7165-53.66]	[1,10]
IoT	$[2, 100]$	$[500, 2000]$	$[512, 2948]$	[87.53-82.44]	[41.22-43.76]	[1,10]

[a]Computation Power, [b]Communication Power.

B. Results

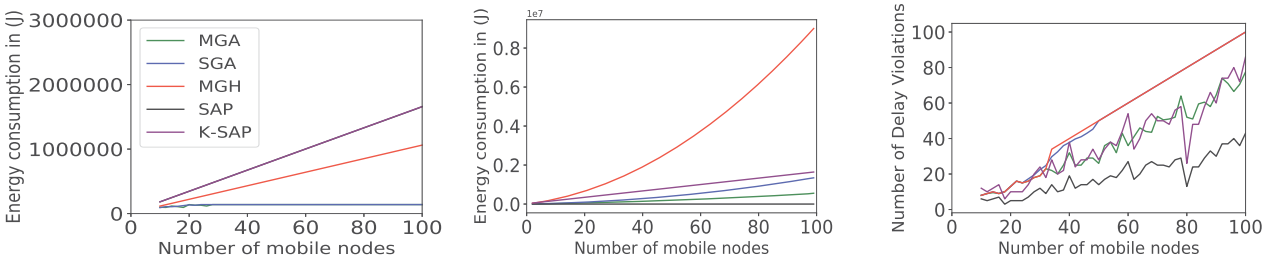
a) Impact of number of IoT nodes on computation energy: Figure 4a plots the average computation energy consumption f_C in joules by increasing number of mobile nodes for each placement method. We observe that both MGA and SGA methods minimize the computation energy consumption and give similar results. The energy consumption due to the services computation is not impacted by nodes mobility: services are still hosted by the same machine during nodes movements. These algorithms give better results than MGH, SAP and K-SAP, the latter two giving similar results which are the worst. The migration process in SAP and K-SAP increases the energy computational cost because it uses VMs replica to ensure services continuity for the IoT nodes. Moreover, those strategies aim at reducing latency between services and mobile nodes and do not consider the energy cost. Also, K-SAP is less efficient compared to MGA and SGA, due to its greedy logic for choosing a host to place a service (it does not reconsider the placement decision for previous placed services).

b) Impact of number of IoT nodes on communication energy: Figure 4b plots the communication energy consumption f_N of the infrastructure in joules by increasing number of mobile nodes for each placement method. In contrast to the previous Figure 4a and as expected, the SAP approach, which migrates the services following nodes paths, gives the lowest communication energy, followed by MGA. By avoiding the migration process, MGA gives as good results as migration strategies while minimizing communication energy consumption. The MGA reduces in average, for all problem size (IoT nodes set size), communication energy consumption by 73%, 55% and 93% compared respectively to K-SAP, SGA and MGH, while SAP outperforms MGA by less than 20%. K-SAP approach is less efficient because it migrates only K nodes services. In this scenario the network cost due to migrations is less important than the network cost due to communications between IoT nodes and their services. SGA method is less efficient to minimize this metric because it takes only the initial position of IoT nodes as input.

c) Impact of number of IoT nodes on delay violations: Figure 4c shows the average number of delay violations f_2 for different number of IoT mobile nodes for each placement approach. We notice that SAP gives the lowest delay, followed by MGA, MGH and SGA (the latter giving similar results). At each slot t, SAP migrates initial services directly on the closest Fog server which minimizes at each slot t the delay between mobile node and its services and reduces in average, for all problem sizes, delay violations by 48% compared to MGA. K-SAP, by migrating only services for half mobile nodes and keeping others on the initial nodes (the closest Fog server), gives worse results. MGA is as good as K-SAP with an average reduction of violations by 0.68% compared to K-SAP, which is due to its probabilistic mobility-based approach for delay violations minimization. MGA also reduces the number of violations by 28% and 29% compared respectively to SGA and MGH.

d) Impact of number of IoT nodes on execution time of the placement method: Figure 5b shows the average computation time of MGA, SGA and MGH methods. MGH is a fast greedy method compared to SGA and MGA which are both evolutionary algorithms and usually those techniques are more time consuming compared to greedy heuristics. Moreover, MGA has an additional computation time compared to SGA which is due to mobile nodes paths estimation. Nevertheless, we notice that MGA takes only 6 seconds to place 270 services (3x90) in an infrastructure containing 90 mobile nodes. In this figure we present the time taken by the placement approaches to choose a services placement combination that minimizes the previously defined objectives. K-SAP and SAP are migration approaches, so their computation time is not comparable with the computation time of the described placement strategies.

e) Results analysis: Figure 5a plots average normalized values within $[0, 1]$ of computation energy, communication energy, delay violations and execution time for all experiments and for each placement method. This Figure comes as a summary to previous presented results. MGA outperforms other placement methods in terms of computation and network energy consumption and gives as good results as K-SAP

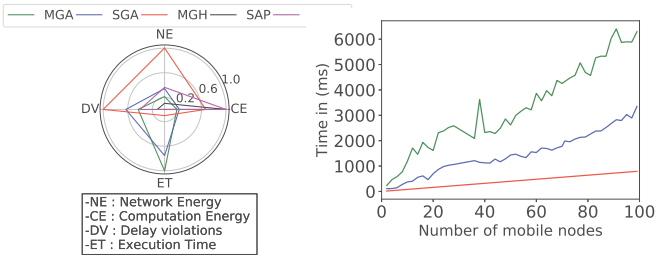


(a) Infrastructure's energy consumption of computation

(b) Infrastructure's energy consumption of communication

(c) Response time delay violations of the applications

Fig. 4. Average energy consumption and delay violations per increasing number of IoT nodes for each placement approach



(a) Normalized metrics values

(b) Execution Time in (ms)

Fig. 5. (a) Normalized average values of computation and network energy consumption, delay violations and execution time for all experiments and methods through radar graph, (b) Average values of execution time per increasing number of IoT nodes for each placement method.

migration strategy to minimize delay violations. From the previous analyzed results, we can conclude that the MGA gives a good trade-off between both energy and delay violations values and gives better results than some migration strategies. The MGA strategy can be considered as a good alternative to avoid migration operations that can be time and energy consuming approaches. MGA offers a new way to handle dynamic environments with mobile nodes.

VI. Conclusion

In this work we have presented a probabilistic mobility-based Genetic Algorithm (MGA) and a mobility greedy heuristic (MGH) for an efficient services placement in the Fog that minimizes the infrastructure energy consumption and applications delay violations over time. We have extended MyiFogSim with mobility models and implemented a pedestrian mobility scenario in a smart campus with a Weighted Waypoints mobility model [16]. The proposed placement method MGA, outperforms the migrations strategies SAP and K-SAP for energy minimization and it outperforms SGA, K-SAP and MGH for delay violations minimization. As future work, we will extend our model with vehicles mobility models and study the scalability of the proposed approaches by increasing the number of application services, since preliminary experiments have shown that it does not impact significantly the approaches performances. We plan to work with various applications topologies to investigate their impact on the placement approaches behaviors.

VII. Acknowledgment

This work is supported by neOCampus project, Toulouse university, France.

References

[1] F. Bonomi, R. Milito, Z. Jiang, A. Sateesh, "Fog Computing and Its Role in the Internet of Things," Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing, Association for Computing Machinery, Finland, 2012, pp. 13–16.

[2] Q. T. Minh, D. T. Nguyen, A. Van Le, H. D. Nguyen and A. Truong, "Toward service placement on Fog computing landscape," 2017 4th NAFOSTED Conference on Information and Computer Science, Hanoi, 2017, pp. 291-296.

[3] O. Skarlat, M. Nardelli, S. Schulte, "Optimized IoT service placement in the fog," Service Oriented Computing and Applications, vol. 11, no. 4, pp. 427–443, 2017.

[4] IEA 4E EDNA, "Energy Efficiency of the Internet of Things," IEA 4E EDNA Technology and Energy Assessment Report, pp. 1–66, 2016.

[5] 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.

[6] A. Mebrek, L. Merghem-Boulahia and M. Esseghir, "Efficient green solution for a balanced energy consumption and delay in the IoT-Fog-Cloud computing," 2017 IEEE 16th International Symposium on Network Computing and Applications (NCA), Cambridge, MA, 2017, pp. 1-4.

[7] L. F. Bittencourt, J. Diaz-Montes, R. Buyya, O. F. Rana and M. Parashar, "Mobility-Aware Application Scheduling in Fog Computing," in IEEE Cloud Computing, vol. 4, no. 2, pp. 26-35, March-April 2017.

[8] T. Djemai, P. Stolf, T. Monteil and J. Pierson, "A Discrete Particle Swarm Optimization Approach for Energy-Efficient IoT Services Placement Over Fog Infrastructures," 2019 18th International Symposium on Parallel and Distributed Computing (ISPDC), Amsterdam, Netherlands, 2019, pp. 32-40.

[9] C. Mouradian, S. Kianpisheh, M. Abu-Lebdeh, F. Ebrahimnezhad, N. T. Jahromi and R. H. Glitho, "Application Component Placement in NFV-Based Hybrid Cloud/Fog Systems With Mobile Fog Nodes," in IEEE Journal on Selected Areas in Communications, vol. 37, no. 5, pp. 1130-1143, May 2019.

[10] R. Cziva, C. Anagnostopoulos and D. P. Pezaros, "Dynamic, Latency-Optimal vNF Placement at the Network Edge," IEEE INFOCOM 2018 - IEEE Conference on Computer Communications, Honolulu, HI, 2018, pp. 693-701.

[11] S. Wang, R. Urgaonkar, M. Zafer, T. He, K. Chan and K. K. Leung, "Dynamic Service Migration in Mobile Edge Computing Based on Markov Decision Process," in IEEE/ACM Transactions on Networking, vol. 27, no. 3, pp. 1272-1288, June 2019.

[12] A. Aissioui, A. Ksentini, A. M. Gueroui and T. Taleb, "On Enabling 5G Automotive Systems Using Follow Me Edge-Cloud Concept," in IEEE Transactions on Vehicular Technology, vol. 67, no. 6, pp. 5302-5316, June 2018.

[13] D. Liu, X. Sui and L. Li, "An energy-efficient virtual machine placement algorithm in cloud data center," 2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), Changsha, 2016, pp. 719-723.

[14] R. Mahmud, B. Rajkumar, "Modelling and Simulation of Fog and Edge Computing Environments using iFogSim Toolkit," ArXiv abs/1812.00994 (2019): n. pag, 2019.

[15] M.M. Lopes, W.A. Higashino, M. Capretz, L.F. Bittencourt, "MyiFogSim: A Simulator for Virtual Machine Migration in Fog Computing," Companion Proceedings of the 10th International Conference on Utility and Cloud Computing, pp. 47–52, 2017.

[16] W.J. Hsu, K. Merchant, H.W. Shu, C.H. Hsu, A. Helmy, "Weighted waypoint mobility model and its impact on ad hoc networks," SIGMOBILE Mob. Comput. Commun, pp. 59–63, 2005.

[17] A. Beloglazov, J. Abawajy, R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for Cloud

computing," Future Generation Computer Systems, vol. 28, no. 5, pp. 755–768, 2012.
[18] N. Chamas, F. López-Pires and B. Baran, "Two-phase virtual machine placement algorithms for cloud computing: An experimental evaluation under uncertainty," 2017 XLIII Latin American Computer Conference (CLEI), Cordoba, 2017, pp. 1-10.
[19] D. Xu, K. Wang, "Stochastic Modeling and Analysis with Energy Optimization for Wireless Sensor Networks," International Journal of Distributed Sensor Networks, vol. 10, no. 5, 2014.
[20] C. Canali, R. Lancellotti, "R. GASP: Genetic Algorithms for Service Placement in Fog Computing Systems," Algorithms, vol. 12, no. 1, pp. 201, 2019.
[21] A. Brogi, S. Forti, C. Guerrero and I. Lera, "Meet Genetic Algorithms in Monte Carlo: Optimised Placement of Multi-Service Applications in the Fog," 2019 IEEE International Conference on Edge Computing (EDGE), Milan, Italy, 2019, pp. 13-17.
[22] M. Nardelli, V. Cardellini, V. Grassi and F. L. Presti, "Efficient Operator Placement for Distributed Data Stream Processing Applications," in IEEE Transactions on Parallel and Distributed Systems, vol. 30, no. 8, pp. 1753-1767, 1 Aug. 2019.
[23] C. Guerrero, I. Lera, C. Juiz, "A lightweight decentralized service placement policy for performance optimization in fog computing," Journal of Ambient Intelligence and Humanized Computing, pp. 1-18, 2018.