Cloudlets Activation Scheme for Scalable Mobile Edge Computing with Transmission Power Control and Virtual Machine Migration

Tiago Gama Rodrigues, *Member, IEEE,* Katsuya Suto, *Member, IEEE,*
Hiroki Nishiyama, *Senior Member, IEEE,* Nei Kato, *Fellow, IEEE,* and Katsuhiro Temma, *Member, IEEE*

Abstract—Mobile devices have several restrictions due to design choices that guarantee their mobility. A way of surpassing such limitations is to utilize cloud servers called cloudlets on the edge of the network through Mobile Edge Computing. However, as the number of clients and devices grows, the service must also increase its scalability in order to guarantee a latency limit and quality threshold. This can be achieved by deploying and activating more cloudlets, but this solution is expensive due to the cost of the physical servers. The best choice is to optimize the resources of the cloudlets through an intelligent choice of configuration that lowers delay and raises scalability. Thus, in this paper we propose an algorithm that utilizes Virtual Machine Migration and Transmission Power Control, together with a mathematical model of delay in Mobile Edge Computing and a heuristic algorithm called Particle Swarm Optimization, to balance the workload between cloudlets and consequently maximize cost-effectiveness. Our proposal is the first to consider simultaneously communication, computation, and migration in our assumed scale and, due to that, manages to outperform other conventional methods in terms of number of serviced users.

Index Terms—Mobile Edge Computing, cloudlet, scalability, resource management, virtualization, Transmission Power Control, Particle Swarm Optimization.

1 Introduction

Mobile devices have inherent limitations that significantly debilitate their capability. This includes shortened battery life, limited communication/computation power, and limited memory. Furthermore, because there will always be a need to maintain portability and mobility, regardless of technological advancements, it is expected that such devices will essentially always lag behind desktop environments in terms of what they can achieve. Consequently, there will always be applications that cannot be executed in purely mobile environments, which is obviously an undesirable limitation of this kind of service [1]. An increasibly popular solution to this issue is Mobile Edge Computing (MEC), which pairs mobile devices with powerful edge servers [2], [3]. Such servers are deployed at the edge of the network to accelerate and facilitate connection; additionally, they are deployed in high numbers, which can be achieved because of their limited capabilities when compared to conventional, backbone cloud servers (which earned them the nickname of cloudlets). In the MEC service model, each user has a corresponding Virtual Machine (VM) server that mimics the environment of their mobile device [2]. This VM server is hosted by a single cloudlet and has access to the resources of the physical edge server. Thus, whenever the mobile device receives a job that is too demanding for it (e.g. it requires too much processing or memory), it sends this job to the cloudlet hosting the corresponding VM server, which will execute it using the resources of the physical cloudlet server and send back the corresponding output to the user. Because the resources of the mobile device are not utilized to execute the job, the aforementioned limitations are not relevant anymore. Because of this, MEC is an effective solution for addressing the shortcomings of mobile environments.

However, there are challenges that must be overcome to properly utilize MEC. The most important requirement of MEC (and cloud computing services in general) is to maintain a degree of transparency [3], i.e., maintain the illusion to the user that all tasks are being executed locally in the mobile device, to a certain degree. This translates to minimizing the latency of the service, which we will call Service Delay. In MEC, we can divide Service Delay into three elements, depending on which part of the service we are referring to. These three elements are Transmission Delay, Processing Delay, and Backhaul Delay. Transmission Delay refers to the time spent transmitting data between the user and the physical cloudlet, i.e., when the user sends the input (the job) to the cloudlet and the cloudlet transmits the output back to the user after the job has finished executing. Processing Delay consists of the time the VM server takes to execute the job sent by the user utilizing the resources of the physical cloudlet, including any necessary queuing for required resources that happen to be busy already. Finally,

- *T. G. Rodrigues, H. Nishiyama, and N. Kato are with the Graduate School of Information Sciences, Tohoku University, Sendai, Miyagi, Japan. E-mail: {tiago.gama.rodrigues, bigtree, kato}@it.is.tohoku.ac.jp*
- *K. Suto is with the University of Waterloo, Waterloo, Ontario, Canada. E-mail: k.suto.jp@ieee.org.*
- *K. Temma is with the National Institute of Information and Communication Technology, Sendai, Miyagi, Japan. E-mail: temma@nict.go.jp.*

Manuscript received ???? ??, 201?; revised ???? ??, 201?; accepted ???? ??, 201?. Date of publication ? ?? 201?; date of current version November 1, 2017.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. ?

the Backhaul Delay comes into effect whenever cloudlets need to communicate amongst each other (e.g., when a user sends the job to a cloudlet that does not host the VM server but offers a better communication environment); this exchange takes place in a backhaul physical link that connects all cloudlets, and the delay is the measure of how long such communication takes to be executed. It is clear that Service Delay is the sum of these three classes of delay. Moreover, because of the need to maintain transparency, it is important for MEC to provide a guaranteed threshold of Service Delay, where users (or even service providers) can expect their jobs to finish under that time limit. This ensures not only maximum latency but also a minimum Quality of Service (QoS). This is specially true for real-time applications [4].

The main obstacle to this requirement is dynamic scenarios where MEC must work. For example, MEC is very attractive in situations where a high number of clients are expected, such as sports events or academic conferences [5]. With MEC, all attendees of such events would have access to powerful cloud computing resources on their mobile devices. However, this type of benefit is accompanied by a need for scalability, because if the cloudlets are overloaded with requests, Service Delay and QoS will break their required thresholds and service transparency will be lifted. Furthermore, not only does the total number of users have to be considered, it is also necessary to cosnider user movement and congregation. There may be moments where the majority of users will move near a fraction of the cloudlets and consequently connect to them, thus overworking those servers while the remaining cloudlets have idle resources [6]. A straightforward solution to both absolute and local scalability is to activate additional cloudlets. However, there are clear problems with this in the form of the sheer economic cost that it entails and the time necessary to identify the problem and activate a new physical server. Thus, we conclude that the best option would be to reconfigure the existing cloudlets and balance the workload between them so that Service Delay can be controlled and minimized (or at the very least kept under the desired threshold) without the need of any additional server activation. Indeed, this solution is preferred in the literature [7].

Therefore, in this paper, we will utilize an analytical model of the Service Delay in MEC (which encompasses Transmission Delay, Processing Delay, and Backhaul Delay) together with Particle Swarm Optimization (PSO, an Artificial Intelligence algorithm [8]) for encountering the configuration that best controls all three elements of Service Delay, manages to minimize latency, and, most importantly, maximize scalability in the form of how many users the system can handle without violating Service Delay/QoS restrictions. To realize and enable the configuration calculated by our PSO algorithm, we utilize Transmission Power Control [9] and VM Migration [10] to manage the workload and parameters of the cloudlets.

2 Related Works on Improving Scalability

As mentioned before, Service Delay (the time between the user producing the task (input) and receiving the corresponding results (output)) is divided into Transmission Delay, Processing Delay, and Backhaul Delay. Because all three components occur in different independent mediums (i.e., the wireless medium between the user and the connected cloudlet, inside the cloudlet that hosts the VM server, and the backhaul link connecting all physical cloudlets), a simple course of action is to deal with them separately by focusing on a single component and trying to more intelligently utilize the resources related to it. This simultaneously lowers the respective delay and increases the respective scalability (i.e., the number of users that can be serviced under a determined latency threshold without activating new cloudlets). Indeed, this strategy of focusing on a single component of the latency can be seen in the literature.

Because all VM servers in the same cloudlet have to share the same resource pool, Processing Delay tends to grow along with the number of VMs contending for such resources (in an effect called multi-tenancy [11], [12]). Thus, if efforts are made to avoid cloudlets hosting more VM servers than necessary, it should be effective in managing Processing Delay and improving computation scalability. As a result, balancing the amount of VM servers hosted by each cloudlet to intelligently use computation resources is often seen in the literature. Roy *et al.* [13], and Oueis *et al.* [14] balanced the workload by always setting up the VM servers of new incoming users in the cloudlet with the least amount of VMs; this guarantees that the difference between the amount of hosted VM servers among the cloudlets is always minimal. Gkatzikis and Koutsopoulos [15], and Mishra *et al.* [16] achieve the same goal by using VM Migration to move VMs between cloudlets. If the migration is done from overworked cloudlets to ones hosting few VMs, then this would improve the efficiency of cloudlet usage by avoiding situations where cloudlets are idle with wasted capacity. Jia *et al.* [17] offer a different insight on VM Migration. The authors point out that too many migrations might be detrimental to Service Delay, even if Processing Delay is lowered, because of the time it takes to perform the migrations themselves and the corresponding usage of the backhaul link (migrations often lead to users needing to send tasks to cloudlets they are not directly connected to). Farrugia [18], and Zhu *et al.* [19] propose a different approach to lowering Processing Delay. Each task would be divided into smaller, independent sub-tasks that would be individually assigned to the cloudlets. Such sub-task assignments take into consideration the estimated completion time before a cloudlet is selected to send the task to, therefore avoiding physical servers that are already too busy and would take longer to finish processing.

Analogous to the approaches focusing on improving computation resources efficiency, there are works in the literature that focus on more intelligently utilizing the resources related to the communication between the user and the connected cloudlet. Because such communication occurs in the wireless medium, these approaches turn to improving metrics related to this channel, such as Signal to Interference Plus Noise Ratio (SINR), Received Signal Strength (RSS) and throughput. Li and Wang [20], and Suto *et al.* [21] propose to always connect users to the cloudlet that is closest to them. In scenarios with homogeneous cloudlets (at least in the communication sense), this would mean that the closest cloudlet offers the highest RSS and best communication conditions. While their assumptions do hold for simple scenarios, it is easy to see how multiple users connecting to the same cloudlet would create congestion that would degrade communication, even if the transmission distance is minimized. Yang*et al.* [22] tackle this issue by not only connecting the user to the closest cloudlet, but also by sending tasks to remote conventional cloud servers if the communication to the former proves to be too troublesome. This is basically using the remote cloud to increase scalability when the edge cloud is not sufficient. However, von Zengen *et al.* [23], and Aota and Higuchi [24] propose utilizing Transmission Power Control to control the individual transmission power levels of each cloudlet in order to diminish the interference they cause between each other (creating a heterogeneous scenario). Because transmission power is tightly connected to SINR, RSS, and channel capacity, control over it also allows for control over most of the wireless medium parameters and Transmission Delay as a whole. Peng *et al.* [25] propose dividing the resources of the physical layer (such as channel bandwidth) between the users to compensate for poor communication conditions by allocating more resources to users in worse situations. This is an efficient way of improving fairness in Transmission Delay.

It is noteworthy that almost all methods for intelligently using cloudlet resources mentioned so far only focus on either Processing Delay or Transmission Delay; they ignore the other components of the latency. This is far from ideal for multiple reasons [26]. Firstly, if all resources (i.e., related to all delays) are not being efficiently utilized, then it is still possible to improve Service Delay and increase scalability. In addition, an approach that focuses solely on Transmission Delay would deliver poor service to an application that relies more on computation (and vice versa). This was also noted in the literature. Mao, Zhang and Letaief [27] utilize Transmission Power Control and task offloading scheduling decision (that is, when and which tasks to send to the cloudlet) to control Transmission Delay and Processing Delay, respectively, when intelligently using the cloudlet, but their models and solutions are applicable to single-user single-cloudlet scenarios only. Sardellitti *et al.* [28] control Processing Delay by determining how many CPU resources to allocate to each user while also controlling Transmission Delay. This is accompanied by selecting how many radio resources to give to each user, but for a multiple-user single-cloudlet scenario only. Chen *et al.* [29], Yu *et al.* [30], and Wang *et al.* [31] also simultaneously consider Transmission Delay and Processing Delay under the same scenario restrictions. Meanwhile, Wang *et al.* [32], Dinh *et al.* [33], and Sato and Fujii [34] all consider the same problem (of improving computation and communication scalability) under a single-user multiple-cloudlets scenario.

Our proposal aims at expanding the limitations of the current literature. We will not focus solely on one element of Service Delay but will simultaneously consider Transmission Delay, Processing Delay, and Backhaul Delay, because this is the most efficient way of improving scalability for all possible application scenarios [35]. Our method of achieving this is to utilize Transmission Power Control to manage the parameters of the wireless medium and the throughput of the communication, and to also use VM Migration to balance the computation workload between physical cloudlets and lower their Processing Delay. Additionally, we will consider the load on the backhaul when determining which and how many migrations to perform. Finally, our assumption is a multiple-user multiple-cloudlet scenario, which is a scale above the scenarios considered by currently existing approaches in the literature that have a hybrid focus on computation and communication. These two points (consideration of all elements of latency and assumption of a greater scenario) lead to a more complex problem to be solved because of the number of extra variables introduced by both considering all types of delay and more users/cloudlets. This complexity is the main difficulty of the problem tackled here, but we address it by analytically modeling the delay and using a customized PSO as a heuristic algorithm.

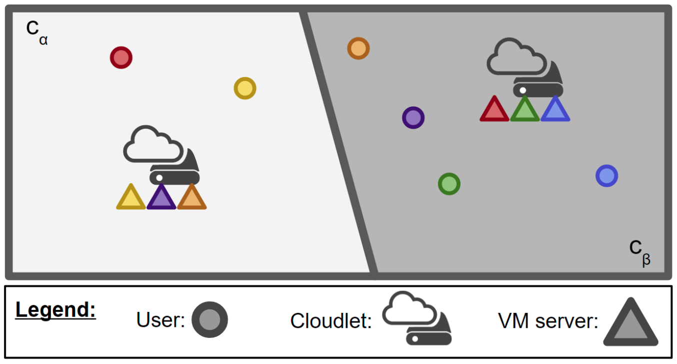


Fig. 1. Illustrative example of six users and two cloudlets and their associations.

Finally, our proposal is focused on MEC and computation offloading to cloudlets, but there are other variations of distributed computing where instead of cloudlets the tasks are offloaded to other user equipments (as in Fog Computing [36]) or Base Station devices [37], for example). Theoretically, our proposal could be applied to those scenarios, but in practice, it would need adaptations to the characteristic features of those situations. Such research is beyond the scope of this work.

3 Analytical Model of MEC Delay

In this section, we will present a mathematical model for calculating the estimated average Service Delay in MEC as a function of time and the scenario scale (i.e., the number of users). We will begin by presenting our assumed scenarios. Then, we will model user mobility and how to calculate the number of users associated with each cloudlet. Finally, we will present equations for calculating Transmission Delay, Processing Delay, and Backhaul Delay before joining the entire model into a calculation of the average Service Delay.

3.1 Assumed Scenario

For starters, we will discuss several assumptions about our scenario. We have a bounded area A (where A denotes the area and its size in square meters). Inside this area, we have the cloudlets and the clients that will be associated with them. $\lambda(X, t_k)$ denotes how many users are in point X at timeslot t_k. Therefore, users are initially disposed in A following a density function $\lambda(X, 0)$. However, users are mobile and can roam from this positions to any point inside of A. Furthermore, to symbolize a growing demand, we have that at timeslot t_k, $\alpha(X, t_k)$ new users appear in point X, where $\alpha(X, t_k)$ is our spawning function. We assume this infinitely increasing workload so we can test our proposal and other methods under extreme stress conditions; if they behave well in this scenario, they are able to handle simpler cases as well [38]. There are $|C|$ cloudlets in the scenario, which all belong to set C; we denote them by c_i, with $0 \leq i < |C|$.

Users have a physical and a virtual association [15]. The physical association determines with which cloudlet they communicate through a direct wireless connection. The user will send their tasks and receive output from this cloudlet. The virtual association determines which cloudlet will host the VM server associated with the user that will actually execute the tasks created and produce the output. If a user is physically associated to cloudlet c_i and virtually associated to cloudlet c_j, then c_i and c_j will communicate through a backhaul link to exchange the task (received at c_i from the user) and the output (produced by c_j and then sent to the user by c_i). We assume there is a switch in the backhaul that connects all cloudlets, and each cloudlet has a physical link connected to this switch [39].

Physical association is decided by RSS, where clients will always associate with the cloudlet that currently offers them the highest RSS value. This means that area A will be divided following a Voronoi diagram between the cloudlets. However, unlike conventional Voronoi diagrams, the subareas are decided based on RSS instead of only Euclidean distance. Therefore, area A is divided into $|C|$ subareas A_{c_i}, where all users inside A_{c_i} are associated with and will send tasks to cloudlet c_i. If a user leaves A_{c_i} and goes into A_{c_j}, then its physical association is changed from c_i to c_j. The virtual association is determined by the initial position of the user only, i.e. the user will virtually associate with the cloudlet that offers the highest RSS at the timeslot the user shows up in the system. Even if the user moves, its virtual association (the host of its VM server) will not change.

As mentioned, $\lambda(X, t_k)$ defines how many total users are in X in timeslot t_k. All these users are physically associated to c_i if $X \in A_{c_i}$, but $\lambda(X, t_k)$ tells us nothing about their virtual association. Therefore, let us define a new density function $\lambda_{c_i}(X, t_k)$ that represents the total number of users virtually associated to cloudlet c_i that are located in X at timeslot t_k. Note how $\lambda_{c_i}(X, t_k) \leq \lambda(X, t_k)$. Fig. 1 illustrates the difference between those two values. Here, A_{c_α} is light gray and A_{c_β} is dark gray. In the light gray area, $\iint \lambda(X, t_k)\mathrm{d}X$ is two because there are two users in that region, the red user and the yellow user, who are represented by circles. Also in the light gray area, $\iint \lambda_{c_\alpha}(X, t_k)\mathrm{d}X$ is one instead of two because, of the two users in that area, only one user (the yellow one) has its VM server (represented by the triangle) hosted by c_α. Analogously, in the dark gray area, $\iint \lambda_{c_\alpha}(X, t_k)\mathrm{d}X$ is two because two users in that area (the purple and orange ones) have their VM servers hosted by c_α; these users are not in the light gray area, i.e., not physically associated to c_α, so they will use the backhaul link and c_β as a relay to reach their VM servers.

If Service Delay increases too much (for example, because of an overload on the backhaul caused by mobility) and goes beyond the acceptable limit, the system enters a Configuration Phase. For example, in our proposal, this means the transmission power levels are changed to create a more efficient cloudlet configuration. Physical and virtual associations are reset to the new resulting RSS Voronoi Diagram, which may lead to VMs being migrated (our proposal is explained more in depth in the next section).

For this paper, we will denote initial moment as t_0, which represents the initial moment of any cloudlet configuration. This can mean either the beginning of the service, or after Virtual Machines are migrated, or after a new cloudlet is activated. Basically, t_0 means the timeslot after a new configuration of transmission power levels is set (with the possibility of VMs having been migrated) and the virtual association of the users follows the RSS Voronoi Diagram. As a final note, it is important to say that some variables here are calculated at each timeslot while others are calculated only once considering the initial user distribution. This was chosen for simplicity, because calculating every single variable for all timeslots would be too complex. The selection of which variables would be considered was based on how susceptible they are to changes in the number of users. For example, variables related to queues vary exponentially with number of users and arrival rate. This variation is also relevant to the migration of users.

3.2 User Mobility

We assume users move thusly [40]: the time dimension is divided into timeslots with equal length $\mathfrak{T}$; at each timeslot t_k, each user chooses a random direction and moves towards it with speed dictated by a probability density function $\zeta()$ (maximum speed is $\mathfrak{v}$). Each direction has an equal chance of being selected, so the probability that a particular direction is picked is $\frac{1}{2\pi}$.

Therefore, the expected value of $\lambda(X, t_k)$ (and $\lambda_{c_i}(X, t_k)$) is determined by how many users ended in point X at the end of timeslot t_k. To calculate such number, we utilize a double integral using the number of users at the previous timeslot, the odds of those users picking the direction of X, and the odds of the users picking the speed that will land exactly into X at the end of the timeslot, over the area dictated by $\Omega(X)$, which is a circle with a radius of $\mathfrak{T}\mathfrak{v}$ and centered in X. Additionally, we must also consider the users that spawned in the point in this timeslot ($\alpha(X, t_k)$). Here, r_X^Y is the Euclidean distance between X and Y, and $\alpha_{c_i}(X, t_k)$ is analogous to $\alpha(X, t_k)$, but considers the spawning of users that are virtually associated to cloudlet c_i only ($\alpha_{c_i}(X, t_k)$ is equal to $\alpha(X, t_k)$ if X is inside the association area of c_i; i.e. the area where c_i offers the highest RSS among all cloudlets, denoted by A_{c_i}; and it is zero otherwise).

$$E[\lambda(X, t_k)] = \alpha(X, t_k) + \iint\limits_{\Omega(X)} \left(\frac{1}{2\pi} \zeta\left(\frac{r_X^Y}{\mathfrak{T}} \right) \lambda(Y, t_{k-1}) \right) \mathrm{d}Y \tag{1}$$

$$E[\lambda_{c_i}(X, t_k)] = \alpha_{c_i}(X, t_k) + \iint\limits_{\Omega(X)} \left(\frac{1}{2\pi} \zeta\left(\frac{r_X^Y}{\mathfrak{T}} \right) \lambda_{c_i}(Y, t_{k-1}) \right) \mathrm{d}Y \tag{2}$$

3.3 Number of Users

As previously mentioned, the number of users in an area A_{c_i} at timeslot t_k is determined by the expected value of $\lambda(X, t_k)$. Therefore, if $U_{A_{c_i}}(t_k)$ is the number of users in area A_{c_i} at timeslot t_k (incidentally also the number of users physically associated to cloudlet c_i at t_k), then we can find this value by calculating the double integral of $\lambda(X, t_k)$ over the desired area.

$$U_{A_{c_i}}(t_k) = \iint\limits_{A_{c_i}} E[\lambda(X, t_k)]\mathrm{d}X \tag{3}$$

The number of virtually associated users, i.e. the number of Virtual Machines receiving new tasks inside of cloudlet c_i, in instant t_k is determined by $V_{c_i}(t_k)$ in an analogous way but using $\lambda_{c_i}(X, t_k)$ instead.

$$V_{c_i}(t_k) = \iint\limits_{A_{c_i}} E[\lambda_{c_i}(X, t_k)]\mathrm{d}X \tag{4}$$

3.4 Transmission Delay

We assume that the channel capacity for transmission follows the Shannon Hartley theorem [41], shown below. Here, $\mathfrak{C}$ is the channel capacity (bits per second), $\mathfrak{B}$ is the channel bandwidth (Hertz), S is the RSS (Watts), N is the Additive White Gaussian Noise [42] spectral density (Watts per Hertz) and I is the sensed interference (Watts).

$$\mathfrak{C} = \mathfrak{B}\log_2\left(1 + \frac{S}{\mathfrak{B}N + I}\right) \tag{5}$$

Additionally, we have the formula for RSS below, for a transmitter a_{TX} and a receiver a_{RX}. The formula calculates the signal power in decibels through the transmission power ($\omega_{a_{TX}}$), the antenna gains at both nodes ($G_{a_{TX}}$ and $G_{a_{RX}}$), the Rayleigh power fading constant (H, which is the value corresponding to 0.5 in the cumulative distribution function for average [43]) and the path loss between both nodes ($L_{a_{\text{RX}}}^{a_{\text{TX}}}$), all given in decibels. It then converts this value to Watts.

$$S_{a_{\text{RX}}}^{a_{\text{TX}}} = 10^{\left(\omega_{a_{\text{TX}}} + G_{a_{\text{TX}}} + G_{a_{\text{RX}}} + H - L_{a_{\text{RX}}}^{a_{\text{TX}}}\right)10}/1000 \tag{6}$$

Path loss is given by the Dual Path Empirical Path Loss model, shown below for a transmitter a_{TX} and a receiver a_{RX}. Here, $r_{a_{\text{RX}}}^{a_{\text{TX}}}$ is the distance between the transmitter and the receiver, n_1 and n_2 are the path loss constants for short and long distance, respectively, and r_b is the breakpoint between both classifications.

$$L_{a_{\text{RX}}}^{a_{\text{TX}}} = L_1 + 10n_1\log_{10}r_{a_{\text{RX}}}^{a_{\text{TX}}} + 10(n_2 - n_1)\left(1 + \frac{r_{a_{\text{RX}}}^{a_{\text{TX}}}}{r_b}\right) \tag{7}$$

The Transmission Delay for each task is given by the average time needed to traverse the distance between user and cloudlet twice (once for downlink and once for uplink), where the average distance for clients of cloudlet c_i is g_{c_i} and the propagation speed is γ; the average time needed to send a packet in the uplink at instant t_k ($D_{c_i}^{\text{up}}(t_k)$); and the average time needed to send a packet in the downlink ($D_{c_i}^{\text{down}}$). Thus, the average Transmission Delay for clients of cloudlet c_i in timeslot t_k is given by

$$\dot{T}_{c_i}(t_k) = 2\frac{g_{c_i}}{\gamma} + D_{c_i}^{\text{up}}(t_k) + D_{c_i}^{\text{down}} \tag{8}$$

In order to find g_{c_i}, we utilize another double integral with the density function, but this time utilizing the distance between the point and cloudlet c_i. Again, because this is an average, we must divide by the total amount of users.

$$g_{c_i} = \frac{\iint\limits_{A_{c_i}} \left(\lambda(X, 0) r_{c_i}^{X}\right) \mathrm{d}X}{\iint\limits_{A_{c_i}} \lambda(X, 0)\mathrm{d}X} \tag{9}$$

In the uplink, we can calculate the average time needed to send a packet through (5) by using the average size of an uplink packet, denoted by p^{up} as shown below. Here, B^{up} is the uplink bandwidth (in Hertz) and I_{c_i} is the interference suffered by cloudlet c_i, which is the receiver used here. Once again, we utilize a double integral to obtain the value for all users in the area and divide the value by the total number of users to find the average.

$$Q_{c_i}^{\text{up}} = \frac{\iint\limits_{A_{c_i}} \left(\lambda(X, 0)\frac{p^{\text{up}}}{B^{\text{up}}\log_2\left(1 + \frac{S_{c_i}^{X}}{B^{\text{up}}N + I_{c_i}}\right)}\right)\mathrm{d}X}{\iint\limits_{A_{c_i}} \lambda(X, 0)\mathrm{d}X} \tag{10}$$

For the uplink, we assume users of the same cloudlet follow round-robin scheduling [44] to send their packets. This eliminates collision between users of the same cloudlet, but still leaves the possibility of interference from users of other cloudlets who may be transmitting at the same time. However, because they also follow round-robin scheduling, only one user from each other cloudlet could cause this interference. To calculate the average interference sensed by c_i, we must calculate the average signal power received from users of each of the other cloudlets and sum these values. This value is found by taking a double integral of the total signal power generated by all users, and dividing by the number of users for average, as shown below.

$$I_{c_i} = \sum_{c_j \in C}^{c_j \neq c_i} \left(\frac{\iint\limits_{A_{c_j}} \left(\lambda(X, 0) S_{c_i}^{X}\right) \mathrm{d}X}{\iint\limits_{A_{c_j}} \lambda(X, 0)\mathrm{d}X}\right) \tag{11}$$

If the users follow a round-robin scheduling, then they follow an M/D/1 queue model [44], [45], where 1 represents the single channel for transmission, D means that the timeslot size is a constant value (τ), and M represents the Poisson process that users follow when generating new packets (tasks) and putting them in the queue while waiting for the channel to be available. The average rate of this process is determined by the average task generation rate for a single user (denoted by Λ_1) multiplied by the number of physical users currently associated to c_i in instant t_k, with the addition of users that did not finish sending their packets at the current timeslot. This last value can be calculated as follows. We first calculate the chance that the user will finish in the current timeslot by dividing the length of the timeslot (τ) by the total time needed ($Q^{\text{up}}_{c_i}$). Because we want the opposite of that (the chance that the user does not finish), we subtract such value from 1; this gives us the rate of users that do not finish per timeslot, so we divide that by the length of the timeslot to obtain the value at users per second. It is noteworthy how this value is only applicable if a single timeslot is not sufficient to send a packet; otherwise, it is ignored. Finally, the formula for the arrival rate of packets at this queue for sending to cloudlet c_i at instant t_k is given by

$$\phi_{c_i}(t_k) = \begin{cases} \Lambda_1 U_{c_i}(t_k) + \frac{1-\frac{\tau}{Q^{\text{up}}_{c_i}}}{\tau} & \text{, if } Q^{\text{up}}_{c_i} > \tau \\ \Lambda_1 U_{c_i}(t_k) & \text{, otherwise.} \end{cases} \tag{12}$$

From this value, the average wait time in the uplink queue for clients of cloudlet c_i at instant t_k follows immediately from queuing theory and is given by

$$\varpi_{c_i}(t_k) = \frac{\phi_{c_i}(t_k)\tau^2}{2(1-\phi_{c_i}(t_k)\tau)} \tag{13}$$

Finally, to calculate the total time spent to send a packet in the uplink, we must first determine how many timeslots are necessary by using $Q^{\text{up}}_{c_i}$ and τ. Because all timeslots involve a waiting time prior to obtaining the channel for the length of the timeslot, we multiply that latter number by the sum of the wait time and τ, which leads us to the formula below.

$$D^{\text{up}}_{c_i}(t_k) = \left\lceil \frac{Q^{\text{up}}_{c_i}}{\tau} \right\rceil (\varpi_{c_i}(t_k) + \tau) \tag{14}$$

For the downlink, we assume that each user is allocated a fraction of the total bandwidth through OFDM [46]. This allows us to concurrently transmit to all users of a single cloudlet without worrying about collision (because the frequency bands are different). We can also utilize this to implement fairness into the model by allocating wider bandwidths to users in less favorable locations (i.e., ones that receive a weaker RSS). To do so, we allocate bandwidth to each user that is proportional to the ratio of the inverse of the logarithm base 2 of the RSS it receives from its cloudlet when compared to the sum of the same value for all users of that cloudlet (this value is obtained through a double integral that involves the density function and the corresponding area). Because we use the inverse, users with higher RSS will get less bandwidth, and vice versa. Thus, assuming that $X_0 \in A_{c_i}$ and B^{down} is the total downlink bandwidth (Hertz), the bandwidth allocated to a user in X_0 is

$$\mathfrak{b}^{\text{down}}_{X_0} = \mathfrak{B}^{\text{down}} \frac{\left(S^{c_i}_{X_0}\right)^{-1}}{\left(\iint\limits_{A_{c_j}} \left(\lambda(X,0) S^{c_i}_{X}\right) \mathrm{d}X\right)^{-1}} \tag{15}$$

Here, although transmissions to clients of the same cloudlet do not interfere with each other, there is still interference coming from the other cloudlets. This occurs because there is no coordination between cloudlets, so they do not allocate the same wavelengths to their users. Therefore, we calculate the interference at that previous user in X_0 associated to c_i by summing the RSS received by the other cloudlets.

$$I_{X_0} = \sum_{c_j \in C}^{c_j \neq c_i} S^{c_j}_{X_0} \tag{16}$$

Finally, the average total time for transmission in the downlink is calculated in the same way as performed in (10), but using the average packet size for downlink (p^{down}), and the new values for bandwidth ((15)) and interference ((16)) instead.

$$D^{\text{down}}_{c_i} = \frac{\iint\limits_{A_{c_i}} \left(\lambda(X,0) \frac{p^{\text{down}}}{\mathfrak{b}^{\text{down}}_X \log_2\left(1+\frac{S^{c_i}_X}{\mathfrak{b}^{\text{down}}_X N + I_X}\right)} \right) \mathrm{d}X}{\iint\limits_{A_{c_i}} \lambda(X,0)\mathrm{d}X} \tag{17}$$

These formulas allow us to calculate the average Transmission Delay at timeslot t_k for all users in the system. This is accomplished by utilizing the average for the users physically associated to each cloudlet, multipliying by the number of users physically associated to that cloudlet, and then dividing the sum of those numbers by the total amount of users for averaging.

$$T_{\text{delay}}(t_k) = \frac{\sum\limits_{c_i \in C} \left(U_{A_{c_i}}(t_k) \dot{T}_{c_i}(t_k) \right)}{\sum\limits_{c_i \in C} U_{A_{c_i}}(t_k)} \tag{18}$$

3.5 Processing Delay

We assume that the access to the processors at each cloudlet follows an M/M/k queue [45], where k is the number of processors for each physical cloudlet, the individual processing time for the tasks comes from a Poisson process with average μ, and tasks arrive following yet another Poisson process. The rate for this former process comes from the average task generation time of a single user (Λ_1) multiplied by the total amount of VM servers hosted by cloudlet c_i at that timeslot ($V_{c_i}(t_k)$). The total arrival rate at timeslot t_k for cloudlet c_i is given by

$$\Lambda_{c_i}(t_k) = V_{c_i}(t_k)\Lambda_1 \tag{19}$$

Queuing theory tells us that the average occupation rate for cloudlet c_i at timeslot t_k can then be derived by

$$\rho_{c_i}(t_k) = \frac{\Lambda_{c_i}(t_k)}{k\mu} \tag{20}$$

From this, the chance of waiting in the processors' queue at cloudlet c_i at instant t_k also comes instantly from queuing theory.

$$\Psi_{c_i}(t_k) = \frac{(k\rho_{c_i}(t_k))^k}{k!} \left((1-\rho_{c_i}(t_k)) \sum_{n=0}^{k-1} \frac{(k\rho_{c_i}(t_k))^n}{n!} + \frac{(k\rho_{c_i}(t_k))^k}{k!} \right)^{-1} \tag{21}$$

Because queuing theory also gives us the average waiting time for these users of cloudlet c_i at instant t_k based on the previous variables (the first factor of the right-hand side in the formula below), we can calculate the average Processing Delay for the clients of cloudlet c_i at timeslot t_k by adding this value to the average time needed to process each task.

$$\dot{P}_{c_i}(t_k) = \Psi_{c_i}(t_k)\frac{1}{1-\rho_{c_i}(t_k)}\frac{1}{k\mu} + \frac{1}{\mu} \tag{22}$$

Finally, the average Processing Delay for all users in the system can be obtained in the same way that (18) was calculated.

$$P_{\text{delay}}(t_k) = \frac{\sum\limits_{c_i \in C}\Big(V_{c_i}(t_k)\dot{P}_{c_i}(t_k)\Big)}{\sum\limits_{c_i \in C} V_{c_i}(t_k)} \tag{23}$$

3.6 Backhaul Delay

As mentioned before, all cloudlets are connected to a backhaul switch [39]. This connection is comprised of $|C|$ fiber optic links (one for each cloudlet, connecting them to the switch), and it is used when a user is physically and virtually associated to two different cloudlets. Let us consider cloudlets c_i and c_j. Through Wavelength-Division Multiplexing (WDM) [47], a dedicated wavelength is reserved in the links connecting c_i and c_j to the switch; this band will be used exclusively for backhaul communications between these two cloudlets, i.e., by users who are physically associated to c_i and virtually associated to c_j or vice-versa. Let us assume this band has a length of $\Re_{c_i,c_j}$ (the order between c_i and c_j is irrelevant here), where, for all pairs of cloudlets in C, $\Re_{c_i,c_j}$ Hz are reserved exclusively for that pair. These $\Re_{c_i,c_j}$ Hz are equally divided between all users (and their VMs) of that pair of cloudlets who need it (users virtually associated to c_i located in A_{c_j} and vice-versa). Thus, we have that, for the pair c_i and c_j, users will have the following bandwidth available to them for backhaul communications during timeslot t_k.

$$\mathfrak{r}_{c_i,c_j}(t_k) = \frac{\Re_{c_i,c_j}}{\iint\limits_{A_{c_i}} E[\lambda_{c_j}(X,t_k)]\mathrm{d}X + \iint\limits_{A_{c_j}} E[\lambda_{c_i}(X,t_k)]\mathrm{d}X} \tag{24}$$

Such bandwidth is utilized both for sending the input packet (from the user to its VM) and the output packet (from the VM to the corresponding user) between both cloudlets. Sending such packets through this dedicated wavelength band is what creates the backhaul delay. Below is a calculation of the average backhaul delay across all users during timeslot t_k. Note how even though the delay is only considered for the users that utilize the backhaul, the division for getting the average considers the total amount of users.

$$B_{\text{delay}}(t_k) = \frac{\sum\limits_{(c_i,c_j)\in C^2}^{c_i \neq c_j}\left(\iint\limits_{A_{c_i}} E[\lambda_{c_j}(X,t_k)]\mathrm{d}X\,\frac{p^{\text{up}}+p^{\text{down}}}{\mathfrak{r}_{c_i,c_j}(t_k)}\right)}{\sum\limits_{c_i\in C} V_{c_i}(t_k)} \tag{25}$$

3.7 Service Delay

Finally, Service Delay can be calculated at each timeslot by adding the delays related to transmission, processing, and the backhaul.

$$S_{\text{delay}}(t_k) = T_{\text{delay}}(t_k) + P_{\text{delay}}(t_k) + B_{\text{delay}}(t_k) \tag{26}$$

4 Cloudlets Activation Scheme

As explained in the previous section, Service Delay can increase and violate its threshold either because of mobility, which may cause cloudlets to be overrun with physical associations and can cause congestion in the backhaul link, or because of the influx of new users, which introduces new workload into the system that the cloudlets may not be able to cope with. A straightforward solution would be to activate a new cloudlet. MEC systems are composed of a limited number of cloudlets, where ideally, for reducing system energy consumption, most cloudlets are in hot-standby or completely turned off. Therefore, by activating one of these deactivated cloudlets, we can introduce new resources to the system and immediately reduce Service Delay. However, each active cloudlet raises energy consumption and service cost; so, in our proposal, we opt to delay this as much as possible by maximizing system scalability (i.e. the number of users that can be serviced without introducing new active cloudlets). Therefore, we introduce a Configuration Phase that occurs each time Service Delay breaks the threshold as an alternative to cloudlet activation to lower Service Delay. In this phase, we use VM Migration [10] and Transmission Power Control [23] to reconfigure the system to maximize scalability. Because VM Migration controls virtual associations (by moving VM servers between cloudlets) and Transmission Power Control determines RSS and physical associations (changing the transmission power of the cloudlets changes which cloudlet produces the most powerful signal for each user), these two techniques are perfect candidates for lowering Service Delay. Only if this reconfiguration is unable to lower Service Delay to below the limit that we activate a new cloudlet at the end of the Configuration Phase.

Fig. 2 shows examples of how Service Delay can grow beyond the threshold as user mobility or new users introduce overwork and congestion. In the figure, (1) represents the regular MEC Service Phase (with users and cloudlets exchanging tasks and results). In (2a), user mobility causes congestion on the right cloudlet (and the backhaul link), which is fixed by our proposed Configuration Phase in (3a) by changing transmission power levels and migrating VMs. In (2b), congestion is caused by a new user, but this is again fixed by our Configuration Phase in (3b). Meanwhile, (3c) and (3d) show how the same issues caused by mobility and new users can be fixed by activating the third cloudlet. The key point of our proposal is to avoid solutions like (3c) and (3d) as much as possible, only resorting to activating a new cloudlet if reconfigurations like (3a) and (3b) are not enough to lower Service Delay to below limit.

The configuration that maximizes scalability is the one that gives the highest output to our Algorithm 1; by delaying as much as possible the next violation of the

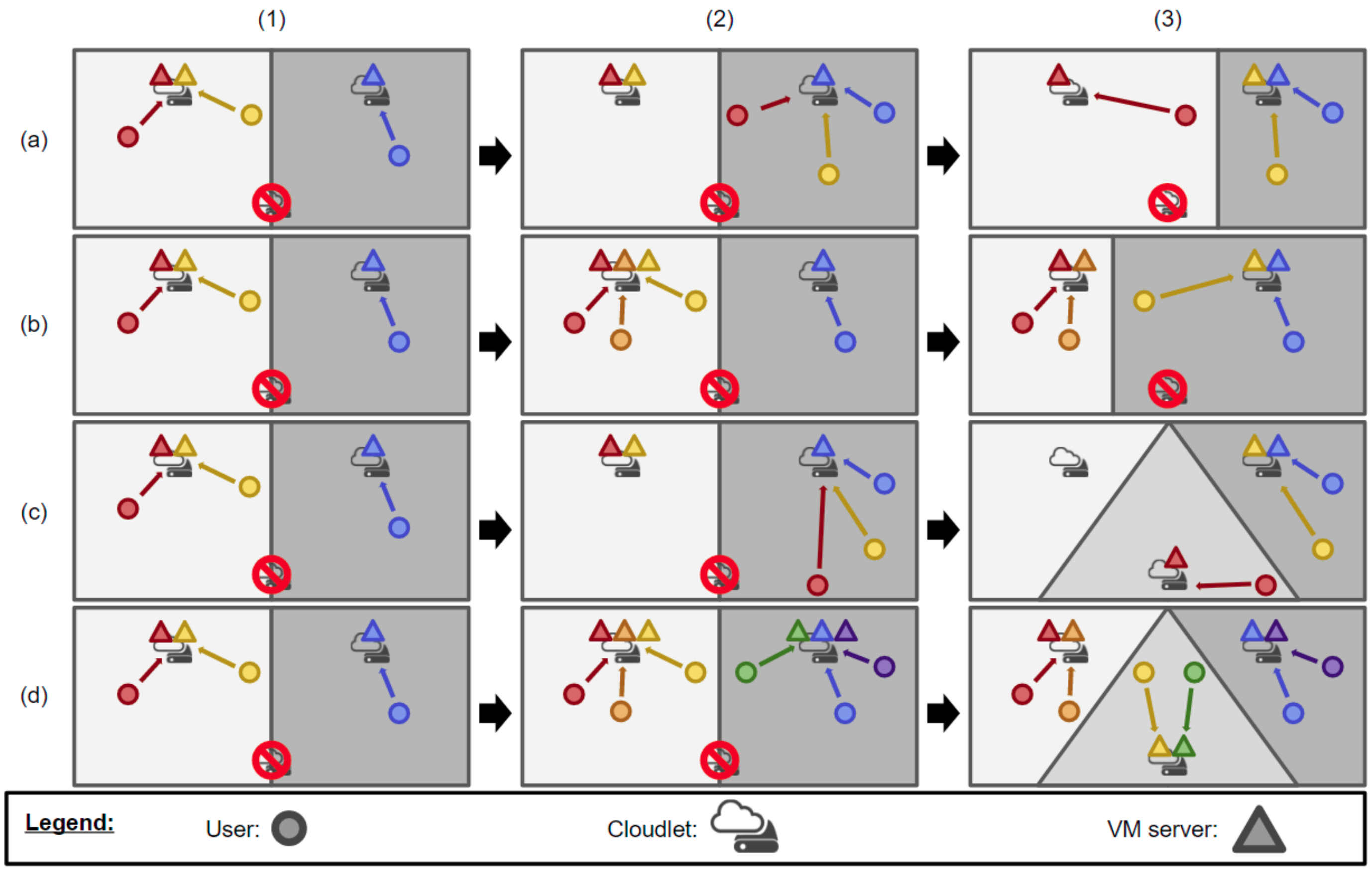


Fig. 2. Illustrative scenarios of how user mobility ((a) and (c)) and new users ((b) and (d)) can increase Service Delay and how cloudlet activation ((a) and (b)) and reconfiguration ((c) and (d)) can counteract this.

Algorithm 1 - NextViolation: estimate next time Service Delay will grow beyond threshold $\mathfrak{H}$

Input: configuration g of transmission power levels of cloudlets, current time t_0

```
1: $\hat{t}_k \leftarrow t_0$
2: while $S_{\text{delay}}(\hat{t}_k) < \mathfrak{H}$ do
3:     $\hat{t}_k \leftarrow \hat{t}_k + 1$
4: return $\hat{t}_k$
```

threshold, we are increasing the number of users that can be served without a new reconfiguration/activation. Note that Service Delay is calculated in line 2 in the algorithm through Equation 26, with input g being utilized in this calculation. We chose transmission power levels as the configuration because both physical and virtual associations are reset in the Configuration Phase to follow RSS levels, which are ultimately decided by the cloudlets transmission power levels. Input g is a tuple with $|C|$ values. However, the problem of finding the input that maximizes the output of Algorithm 1 has $|C|$ decision variables, making it too complex to be solved by conventional methods (e.g., mathematically or through Linear Programming) for high amounts of cloudlets. Thus, because an optimal solution is unreachable in a feasible time, we opted to utilize the Artificial Intelligence algorithm PSO to find a near-optimal configuration.

Algorithm 2 - PSO: algorithm for maximizing MEC scalability

Input: current time t_0

```
1: for all $m \in M$ do initialize $q_m$ as a random $|C|$-tuple
2: for all $m \in M$ do initialize $v_m$ as a random $|C|$-tuple
3: while executed iterations $< L$ do
4:     for all $m \in M$ do
5:         $\hat{t}_k \leftarrow NextViolation(q_m, t_0)$
6:         if $\hat{t}_k > NextViolation(h_m, t_0)$ then $h_m \leftarrow q_m$
7:         if $\hat{t}_k > NextViolation(g, t_0)$ then $g \leftarrow q_m$
8:         $\varphi_h \leftarrow randomInt(0, 1)$
9:         $\varphi_g \leftarrow randomInt(0, 1)$
10:        $v_m \leftarrow \vartheta v_m + \varphi_h \varrho_h (q_m - h_m) + \varphi_g \varrho_g (q_m - g)$
11:        $q_m \leftarrow q_m + v_m$
12: return $(g, NextViolation(g, t_0))$
```

PSO [8] is an algorithm where particles travel through the search space of all possible solutions looking to maximize the value of a chosen fitness function; each particle represents a possible solution to the problem. After each iteration, the particles update their positions (and therefore their corresponding solutions) based on their velocities, which are themselves updated at the end of iterations based on three components: the previous velocity, the best solution found so far locally (by that particle), and the best solu-

Algorithm 3 - ProposedCloudletsActivationScheme: using PSO for maximizing scalability

```
1: repeat
2:     $t_0 \leftarrow$ current time
3:     $(g, \hat{t}_k) \leftarrow PSO(t_0)$                ▷ Configuration Phase
4:     if $\hat{t}_k > t_0$ then
5:         configure cloudlets following $g$
6:         calculate new physical and virtual associations
7:         while Service Delay is below threshold do
8:             continue                              ▷ Service Phase
9:     else
10:        randomly activate one inactive cloudlet
11: until no more cloudlets left to activate
```

tion found so far globally (by all particles). Each of these components has a corresponding weight that determines the degree to which they can influence the velocity. The inertia component (ϑ), local acceleration (ϱ_h), and global acceleration (ϱ_g) are the corresponding weights. Because the velocity is influenced by the best solutions found, the particles tend to move towards better solutions at each iteration [48]. Furthermore, the initial location and velocity of each particle are set randomly to allow for a thorough search of the space (another way of improving this is adding a chance of ignoring local and global best solutions when updating the velocity through binary weights ϱ_h and ϱ_g). For our problem specifically, the search space is composed of all possible combinations of possible values of transmission power levels for the cloudlets (thus, a $|C|$-dimensional space) and the fitness function is defined by Algorithm 1. The PSO algorithm we utilize is shown in pseudo-code in Algorithm (2), where L is the number of iterations, M is the set of particles, and q_r and v_r represent their position and velocity, respectively.

In summary, we propose to utilize the PSO algorithm depicted in Algorithm 2 at each Configuration Phase of the Service Model to calculate a new configuration (i.e. transmission power levels for the cloudlets and the resulting physical/virtual associations of the users). This procedure is shown in Algorithm 3, where the Service Phase is the regular behavior of MEC, with users sending tasks and cloudlets responding with results. Our proposed scheme only resorts to cloudlet activation when a reconfiguration through PSO is unable to lower Service Delay below the threshold, i.e. Algorithm 2 returns as second output that it cannot delay Configuration Phase beyond the current time.

Because of our choice of Algorithm 1 as a fitness function, the configuration found by the particles maximizes scalability and avoids cloudlet activation. Furthermore, because line 2 in Algorithm 1 takes into account Transmission Delay, Processing Delay, and Backhaul Delay, our proposal considers all elements of Service Delay. Regarding scalability, it is capable of handling multiple servers and multiple users; the algorithm efficiency worsens with number of cloudlets (which increases the search space), but it still can be used in large scale scenarios (i.e. a Metropolitan Area Network with tens of servers) with good performance (as shown in the next section) through PSO. These two points are in contrast to the conventional methods from the literature presented in Section 2.

TABLE 1
Simulation Parameters

Application	A	B	C
Average packet size	1.5MB	1MB	2MB
Average task service time	550ms	600ms	500ms
User initial density	1000 users/km^2		
User arrival rate	1 user/(km^2*s)		
Total area size	0.01 km^2		
User maximum speed	3 m/s		
User average speed	1.5 m/s		
User speed variance	0.25 m/s		
Bandwidth (up and downlink)	1GHz		
User transmission power	27dBm		
User device total gain	8.35dBi		
Cloudlet total gain	24.5dBi		
Noise spectral density	4*10^{-19}W/Hz		
Short distance path loss coefficient	2		
Long distance path loss coefficient	4		
Path loss distance breakpoint	100m		
Wireless propagation speed	3*10^{8}m/s		
Processors per cloudlet	16		
Single user task arrival rate	6 tasks/min		
Round robin timeslot	85ms		
Service Delay threshold	2.5s		
PSO particles	5		
PSO iterations	20		

5 Performance Evaluation

In this section, we will evaluate the performance of our proposal. Such evaluation will be done both to optimize the PSO algorithm to our MEC scalability maximization problem and also to compare the proposal with other conventional methods to demonstrate how considering Transmission Delay, Processing Delay and Backhaul Delay simultaneously is significantly better and can handle a higher variety of applications when compared to focusing solely on one of those elements.

5.1 Assumed Scenario

Table 1 contains the parameters utilized in all simulations results shown in this section. The results are obtained from stochastic simulations based on the analytical model presented in Section 3. We assume that users constantly enter the system following a Poisson distribution of inter-arrival times; additionally, once a user has joined the system, we assume it will never leave, in an extreme case increasing workload scenario [38]. These two assumptions guarantee that our system will have an increasing workload with time that must be handled by the cloudlets to keep Service Delay below the acceptable threshold. If such restriction is violated, then the system will first attempt to reconfigure itself to lower Service Delay. The meaning of reconfiguration varies depending on the method being used, but in the case of our proposal it means utilizing Algorithm 3. If the Service Delay levels are above what is allowed even after

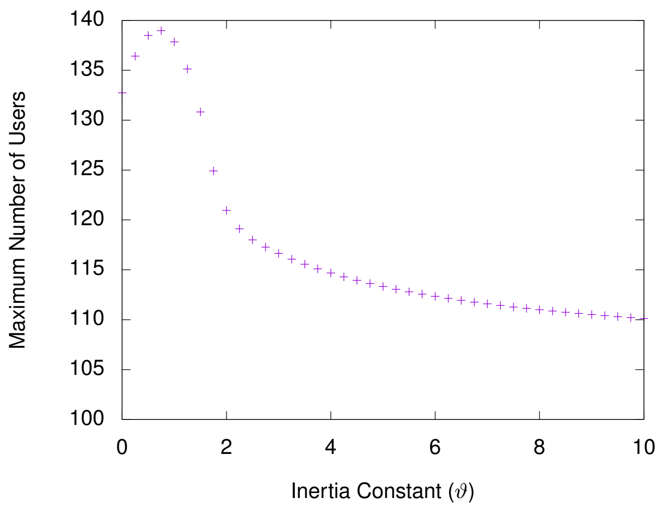


Fig. 3. Maximum number of users served by our proposal with five cloudlets under different values for PSO local inertia (ϑ).

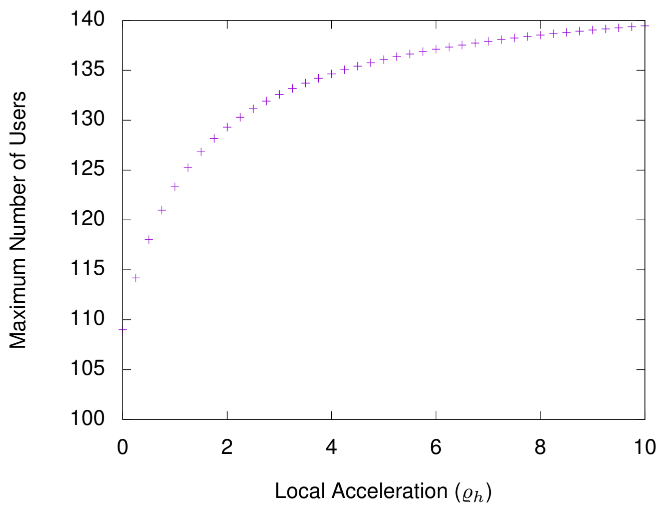


Fig. 4. Maximum number of users served by our proposal with five cloudlets under different values for PSO local acceleration (ϱ_h).

reconfiguration, then the system will activate a new cloudlet server. For all simulations, there is a maximum number of cloudlet servers. We evaluate the proposal based on how many users can be served with the limited number of cloudlets while simultaneously respecting the Service Delay threshold. Obviously, serving more users with this guaranteed latency (service quality) means a higher scalability.

The results shown in this section come from an average of 50 different random runs for added confidence, where each run has a different random seed for its distributions and different random locations for the cloudlets. Application A denotes an average application, Application B requires more processing (i.e. the tasks have a higher average of needed service time) and Application C requires more transmission (i.e. the tasks send bigger packets both in input and output). User mobility follows a normal distribution while the user arrival location follows a uniform distribution.

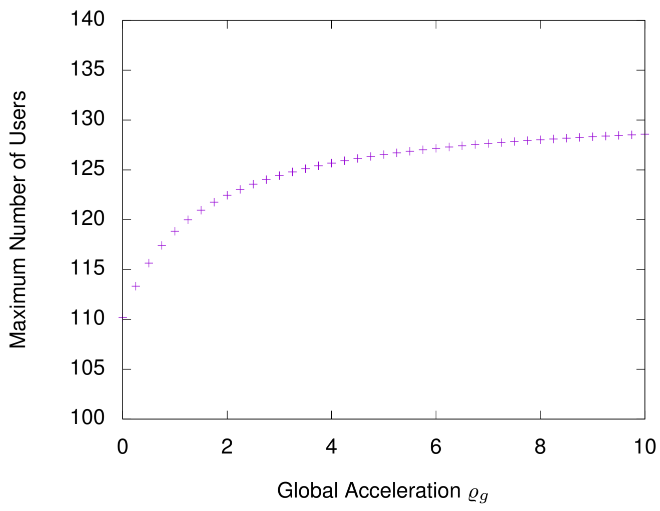


Fig. 5. Maximum number of users served by our proposal with five cloudlets under different values for PSO global acceleration (ϱ_g).

5.2 Hyper-Parameter Analysis

We begin by analyzing the behavior of the fitness function with different configurations of PSO hyper-parameters. This is important because PSO is a general algorithm that we can tune to our respective problem. By doing this, we can find better results in fewer iterations and with fewer particles. This analysis will be done through variation of the value of the parameters and evaluation of how many users can be served while respecting the threshold. For this section, we work with five cloudlets initially activated and no further activations are possible, which gives us the achievable scalability of our proposal with five servers under different PSO hyper-parameter configurations. By analyzing which values give us better results, we can profile our fitness function, which would not be possible through conventional, pure mathematical methods due to its complexity.

The first parameter is the local inertia. It determines the weight of the previous iteration's velocity when updating the speed of the particle. Because the velocity is set as random initially, higher values of local inertia lead to a more random search in the domain of possible solutions. However, as Fig. 3 shows, for our MEC scalability problem, low values of inertia are preferred (i.e. achieve higher scalability and serve more users). This points to an inclination towards exploitation of already found best results over random exploration of the search space [49]. The lower the inertia, the less relevant the random initial speed, leading the particles to rely more on the local best and the global best-found solutions. Since a thorough random search of the space is not necessary, we can deduce that there are not many maximum points in our function, so that it is better to exploit the area near the maximum points already discovered than to waste time sweeping through the search space looking for a better maximum point.

Then we analyze the results achieved with different values of local acceleration and global acceleration. These parameters decide the weight given to the best solution found by that particular particle and all particles respectively. The results are shown in Fig. 4 and Fig. 5. Both graphs show a preference for high values, which corroborates our previous conclusion that exploitation is better than exploration for our MEC scalability function: higher acceleration values mean a higher focus on already found maximum points over a random search [50]. We can also deduce from this preference for exploitation that our maximum points and optimum point are not only few in number (which lowers the chance of the particle getting stuck in a local and not global maximum) but also surrounded by a wide area of constant increase that should be exploited more to find the actual maximum point.

In summary, from these insights, we can conclude that exploitation is better than exploration, there are few maximum points, and they are surrounded by wide areas of constant increase. Therefore, for the rest of the section, we use 0.7 for local inertia (which is low enough without completely ignoring exploration) and 8 for both global and local acceleration.

5.3 Application Profile Analysis

Additionally, we compare the performance of our proposal with other conventional methods from the literature. First we have the No Migration method [20], where users always stay virtually associated to the same cloudlet (the one that offered the highest RSS at the previous Configuration Phase). With no migration or reconfiguration being executed, this is the simplest method and it is used as a benchmark (what can be achieved by doing nothing). The second method is the Minimum Processing method [15], where VM servers are migrated at all Configuration Phases and everytime a user joins the service. This guarantees that the computation workload is perfectly balanced. In addition, it minimizes Processing Delay by using the processor resources of all cloudlets as efficiently as possible, but it also ignores Backhaul Delay (migrations are done with no regards to the cost of transmission between cloudlets) and Transmission Delay (cloudlets may perfectly share virtual associations, but no control is done on physical associations). Finally, we have the Minimum Flow method [17], where at each Configuration Phase, the cloudlets perform just enough migrations between themselves so that the decrease in Processing Delay is enough to keep the average Service Delay under the determined threshold. Because the minimum number of necessary migrations is performed, this minimizes the load in the backhaul link, but it still does nothing to control physical association and Transmission Delay. In addition, the Minimum Flow only activates a new cloudlet if the computing workload is already perfectly balanced, but latency is still too high. The performance evaluation of these three conventional methods and our proposal is done in scenarios that begin with one active cloudlet and allow up to ten active cloudlets. Additionally, we analyze the results with three different types of applications: A (which has no higher leaning towards either transmission or processing), B (which needs more processing in the form of higher time needed to execute the tasks) and C (which has bigger packets to be sent both in downlink and uplink, therefore requiring more attention to transmission).

Fig. 6 contains the performance of all methods in terms of what the probability is that the latency is under the

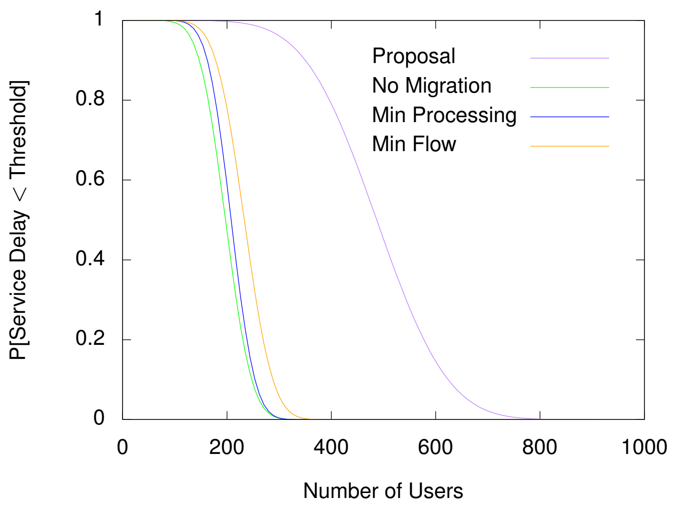


Fig. 6. Probability that latency is below the threshold based on the number of users using Application A (no relevant difference between communication or computation burdens).

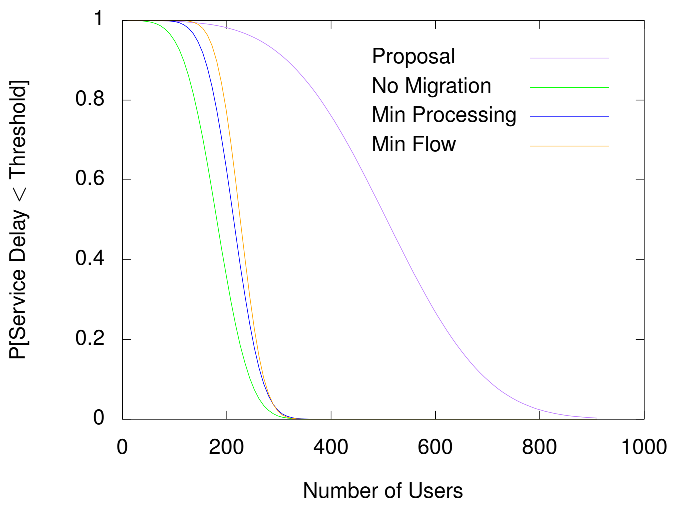


Fig. 7. Probability that latency is below the threshold given the number of users using Application B (heavier burden on processing).

threshold with 10 or fewer cloudlets activated, while utilizing Application A as a service. We can see here how the Minimum Processing method, despite minimizing Processing Delay, behaves almost as bad as the No Migration method because of the high load it introduces to the backhaul link with its many migrations. Meanwhile, the Minimum Flow method is capable of outperforming both due to its consideration of both Backhaul Delay and Processing Delay; however, the proposed method is the best at staying below the latency limit because it is the only method that considers communication and control physical association with Transmission Power Control. Because our method focuses on using as few cloudlets as possible, it aims at delaying reconfigurations and cloudlet activations, which ends up increasing the number of users served per cloudlet.

As seen in Fig. 7, because of the higher burden on computation from Application B, the Minimum Processing

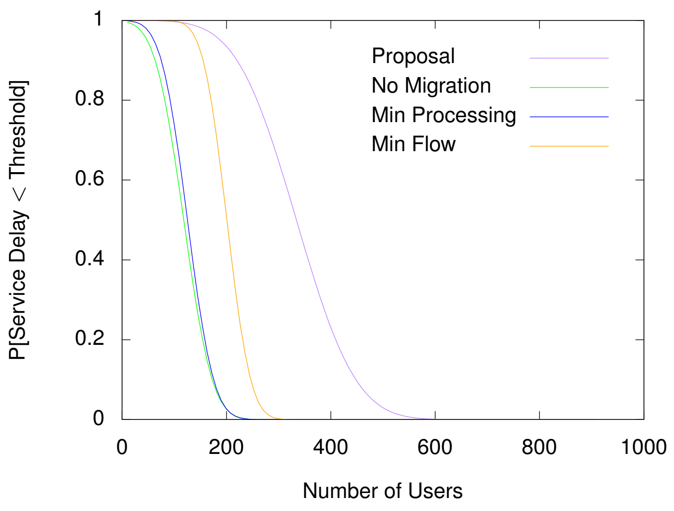


Fig. 8. Probability that latency is below the threshold given the number of users using Application C (heavier burden on transmission).

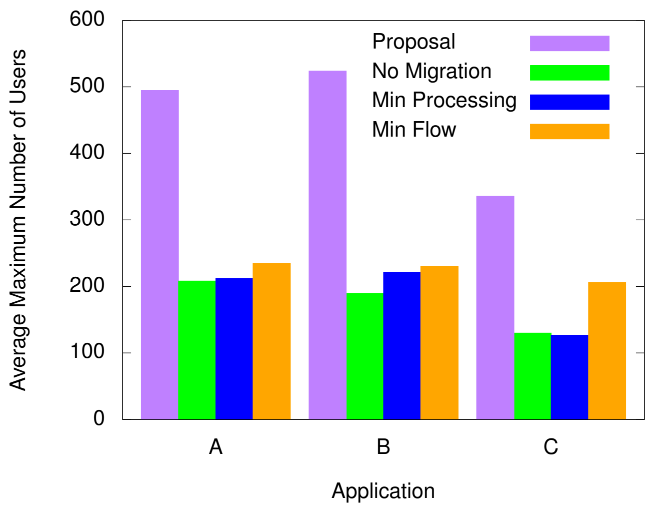


Fig. 9. Maximum number of users for all methods under all applications.

method has better relative performance when compared to the Minimum Flow method because the former minimizes Processing Delay. However, it is still worse than the latter because it ignores Backhaul Delay. Analogously, both methods are still outperformed by our proposal because we are the only one to consider Transmission Delay, which affords us greater control over latency and to serve more users, even in scenarios where computation is more relevant than communication. In fact, Transmission Delay is very important even in this kind of scenario. Which is why the proposal is capable of serving many more users, as evidenced by how early in comparison the curves for the conventional methods end.

Finally, we measure the performance under Application C, which requires the users and the cloudlets to send bigger packets, which demands more transmission time. This higher importance of the Transmission Delay and Backhaul Delay means the Minimum Processing method has its worse performance yet when compared to the Minimum Flow method, as evidenced by Fig. 8. This can be explained by the high burden that the former introduces in the link between cloudlets, whose effect is accentuated by the bigger packets being sent in such link. Nonetheless, the proposal is still the one to offer the best chances of having the threshold respected because it is the only one to truly consider Transmission Delay.

The advantages brought by our proposal are more clearly seen in Fig. 9, where it can be noted how we are able to serve up to three times more users when compared with the conventional methods. We can also see here how the difference between Minimum Processing and Minimum Flow is the smallest when computation is more relevant (Application B) and the biggest when it is less relevant (Application C). In addition, it is obvious how all methods behave worse when communication is more significant, which points to higher difficulties overall with dealing with Transmission Delay (even for our proposal). This only strenghtens our point that considering Transmission Delay is essential both when lowering Service Delay and when maximizing scalability.

6 Conclusion and Future Works

MEC is a method for delivering powerful cloud resources to users on the edge of the network. For this kind of service, the delay must be kept under a threshold to maintain a high-quality experience and transparency. In high workload scenarios, this can be achieved by activating new cloudlets or reconfiguring and optimizing the already active ones, where the former is preferred due to its economy and scalability.

In this paper, we presented a method for reconfiguring cloudlets with the goal of maximum scalability. Our proposal is the first to consider all aspects of the Service Delay in MEC (namely Processing Delay, Transmission Delay and Backhaul Delay) in a multi-user multi-cloudlet scenario. We have tuned PSO for our proposal by a hyper-parameter study that proved that, in the MEC scalability problem, exploitation is better than exploration on the search space of possible solutions when looking for the optimal configuration. Additionally, we compared our proposal with other methods that consider only a fraction of the elements of Service Delay and proved how we can serve more users given a latency limit by considering all elements simultaneously. Furthermore, the same high-quality results can be achieved under different application profiles, proving how our proposal is capable of providing a scalable service in scenarios with different types of requirements.

For the future, we plan to study the cloudlet deployment problem as well as the possible cooperation between cloudlets and backbone cloud servers and the use of wireless communication in the backhaul link between cloudlets.

Acknowledgments

The research results present in this paper have been achieved by Research and Development on Intellectual ICT System for Disaster Response and Recovery, the Commissioned Research of the National Institute of Information and Communications Technology (NICT), Japan.

REFERENCES

[1] M. Satyanarayanan, "Fundamental Challenges in Mobile Computing," in *Proceedings of the 15th Annual ACM Symposium on Principles of Distributed Computing*, May 1996, pp. 1–7.

[2] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The Case for VM-Based Cloudlets in Mobile Computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, October 2009.

[3] A. Ahmed and E. Ahmed, "A survey on mobile edge computing," in *Proceedings of the 2016 10th International Conference on Intelligent Systems and Control (ISCO)*, January 2016, pp. 1–8.

[4] N. Fernando, S. W. Loke, and W. Rahayu, "Mobile cloud computing: A survey," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 84 – 106, January 2013.

[5] S. Tang, X. Li, X. Huang, Y. Xiang, and L. Xu, "Achieving Simple, Secure and Efficient Hierarchical Access Control in Cloud Computing," *IEEE Transactions on Computers*, vol. 65, no. 7, pp. 2325–2331, July 2016.

[6] D. Puthal, B. P. S. Sahoo, S. Mishra, and S. Swain, "Cloud Computing Features, Issues, and Challenges: A Big Picture," in *Proceedings of the 2015 International Conference on Computational Intelligence and Networks (CINE)*, January 2015, pp. 116–123.

[7] T. G. Rodrigues, K. Suto, H. Nishiyama, and N. Kato, "A PSO model with VM migration and transmission power control for low Service Delay in the multiple cloudlets ECC scenario," in *Proceedings of the 2017 IEEE International Conference on Communications (ICC)*, May 2017, pp. 1–6.

[8] M. R. Bonyadi and Z. Michalewicz, "Particle Swarm Optimization for Single Objective Continuous Space Problems: A Review," *Evolutionary Computation*, vol. 25, no. 1, pp. 1–54, March 2016.

[9] Z. Niu, Y. Wu, J. Gong, and Z. Yang, "Cell zooming for cost-efficient green cellular networks," *IEEE Communications Magazine*, vol. 48, no. 11, pp. 74–79, November 2010.

[10] A. Ceselli, M. Premoli, and S. Secci, "Mobile Edge Cloud Network Design Optimization," *IEEE/ACM Transactions on Networking*, vol. 25, no. 3, pp. 379–423, June 2017.

[11] R. Krebs, C. Momm, and S. Kounev, "Architectural Concerns in Multi-Tenant SaaS Applications," in *Proceedings of the 2012 2nd International Conference on Cloud Computing and Services Science (CLOSER)*, April 2012.

[12] J. Duan and Y. Yang, "A Load Balancing and Multi-Tenancy Oriented Data Center Virtualization Framework," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 8, pp. 2131–2144, August 2017.

[13] S. Roy, R. Bose, and D. Sarddar, "Fuzzy based dynamic load balancing scheme for efficient edge server selection in Cloud-oriented content delivery network using Voronoi diagram," in *Proceedings of the 2015 IEEE International Advance Computing Conference (IACC)*, June 2015, pp. 828–833.

[14] J. Oueis, E. C. Strinati, and S. Barbarossa, "The Fog Balancing: Load Distribution for Small Cell Cloud Computing," in *Proceedings of the 2015 IEEE 81st Vehicular Technology Conference (VTC Spring)*, May 2015, pp. 1–6.

[15] L. Gkatzikis and I. Koutsopoulos, "Migrate or not? Exploiting dynamic task migration in mobile cloud computing systems," *IEEE Wireless Communications*, vol. 20, no. 3, pp. 24–32, June 2013.

[16] M. Mishra, A. Das, P. Kulkarni, and A. Sahoo, "Dynamic resource management using virtual machine migrations," *IEEE Communications Magazine*, vol. 50, no. 9, pp. 34–40, September 2012.

[17] M. Jia, W. Liang, Z. Xu, and M. Huang, "Cloudlet Load Balancing in Wireless Metropolitan Networks," in *Proceedings of the 2016 IEEE 35th International Conference on Computer Communications (INFOCOM)*, April 2016, pp. 1–9.

[18] S. Farrugia, "Mobile Cloud Computing Techniques for Extending Computation and Resources in Mobile Devices," in *Proceedings of the 2016 4th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*, March 2016, pp. 1–10.

[19] X. Zhu, C. Chen, L. T. Yang, and Y. Xiang, "ANGEL: Agent-Based Scheduling for Real-Time Tasks in Virtualized Clouds," *IEEE Transactions on Computers*, vol. 64, no. 12, pp. 3389–3403, December 2015.

[20] Y. Li and W. Wang, "The unheralded power of cloudlet computing in the vicinity of mobile devices," in *Proceedings of the 2013 IEEE Global Communications Conference (GLOBECOM)*, December 2013, pp. 4994–4999.

[21] K. Suto, K. Miyanabe, H. Nishiyama, N. Kato, H. Ujikawa, and K. I. Suzuki, "QoE-Guaranteed and Power-Efficient Network Operation for Cloud Radio Access Network With Power Over Fiber," *IEEE Transactions on Computational Social Systems*, vol. 2, no. 4, pp. 127–136, December 2015.

[22] L. Yang, J. Cao, G. Liang, and X. Han, "Cost Aware Service Placement and Load Dispatching in Mobile Cloud Systems," *IEEE Transactions on Computers*, vol. 65, no. 5, pp. 1440–1452, May 2016.

[23] G. von Zengen, F. Bsching, W. B. Pttner, and L. Wolf, "Transmission power control for interference minimization in WSNs," in *Proceedings of the 2014 International Wireless Communications and Mobile Computing Conference (IWCMC)*, August 2014, pp. 74–79.

[24] T. Aota and K. Higuchi, "A simple downlink transmission power control method for worst user throughput maximization in heterogeneous networks," in *Proceedings of the 2013 7th International Conference on Signal Processing and Communication Systems (ICSPCS)*, December 2013, pp. 1–6.

[25] M. Peng, K. Zhang, J. Jiang, J. Wang, and W. Wang, "Energy-Efficient Resource Assignment and Power Allocation in Heterogeneous Cloud Radio Access Networks," *IEEE Transactions on Vehicular Technology*, vol. 64, no. 11, pp. 5275–5287, November 2015.

[26] T. G. Rodrigues, K. Suto, H. Nishiyama, N. Kato, K. Mizutani, T. Inoue, and O. Akashi, "Towards a Low-Delay Edge Cloud Computing through a Combined Communication and Computation Approach," in *Proceedings of the 2016 84th IEEE Vehicular Technology Conference (VTC-Fall)*, September 2016, pp. 1–5.

[27] Y. Mao, J. Zhang, and K. B. Letaief, "Joint Task Offloading Scheduling and Transmit Power Allocation for Mobile-Edge Computing Systems," in *Proceedings of the 2017 IEEE Wireless Communications and Networking Conference (WCNC)*, March 2017, pp. 1–6.

[28] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint Optimization of Radio and Computational Resources for Multicell Mobile-Edge Computing," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 1, no. 2, pp. 89–103, June 2015.

[29] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient Multi-User Computation Offloading for Mobile-Edge Cloud Computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795–2808, October 2016.

[30] Y. Yu, J. Zhang, and K. B. Letaief, "Joint Subcarrier and CPU Time Allocation for Mobile Edge Computing," in *Proceedings of the 2016 IEEE Global Communications Conference (GLOBECOM)*, December 2016, pp. 1–6.

[31] C. Wang, F. R. Yu, C. Liang, Q. Chen, and L. Tang, "Joint Computation Offloading and Interference Managementin Wireless Cellular Network with Mobile Edge Computing," *IEEE/ACM Transactions on Vehicular Technology*, vol. 66, no. 8, pp. 7432–7445, August 2017.

[32] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-Edge Computing: Partial Computation Offloading Using Dynamic Voltage Scaling," *IEEE Transactions on Communications*, vol. 64, no. 10, pp. 4268–4282, October 2016.

[33] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. S. Quek, "Offloading in Mobile Edge Computing: Task Allocation and Computational Frequency Scaling," *IEEE Transactions on Communications*, vol. 65, no. 8, pp. 3571–3584, August 2017.

[34] K. Sato and T. Fujii, "Measuring exploration/exploitation in particle swarms using swarm diversity," in *Proceedings of the 2017 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, March 2008, pp. 1128–1134.

[35] T. G. Rodrigues, K. Suto, H. Nishiyama, and N. Kato, "Hybrid Method for Minimizing Service Delay in Edge Cloud Computing Through VM Migration and Transmission Power Control," *IEEE Transactions on Computers*, vol. 66, no. 5, pp. 810–819, May 2017.

[36] I. Bisio, F. Lavagetto, C. Garibotto, A. Sciarrone, T. Penner, and M. Guirguis, "Context-Awareness Over Transient Cloud in D2D Networks: Energy Performance Analysis and Evaluation," *Transactions on Emerging Telecommunications Technologies*, vol. 28, no. 2, February 2017.

[37] S. Zhang, P. He, K. Suto, P. Yang, and X. S. Shen, "Cooperative Edge Caching in User-Centric Clustered Mobile Networks," *IEEE Transactions on Mobile Computing*, December 2017.

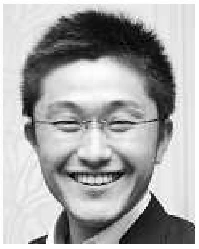
[38] M. Y. Nam, J. Lee, K. J. Park, L. Sha, and K. Kang, "Guaranteeing the End-to-End Latency of an IMA System with an Increasing Workload," *IEEE Transactions on Computers*, vol. 63, no. 6, pp. 1460–1473, June 2014.

[39] Y. J. Yu, T. C. Chiu, A. C. Pang, M. F. Chen, and J. Liu, "Virtual machine placement for backhaul traffic minimization in fog radio access networks," in *Proceedings of the 2017 IEEE International Conference on Communications (ICC)*, May 2017, pp. 1–7.

[40] C. Bettstetter, G. Resta, and P. Santi, "The node distribution of the random waypoint mobility model for wireless ad hoc networks," *IEEE Transactions on Mobile Computing*, vol. 2, no. 3, pp. 257–269, July 2003.
[41] C. E. Shannon, "A Mathematical Theory of Communication," *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.
[42] K. McClaning and T. Vito, *Radio Receiver Design*. Noble Publishing Corporation, 2000.
[43] B. Sklar, "Rayleigh fading channels in mobile digital communication systems. I. Characterization," *IEEE Communications Magazine*, vol. 35, no. 7, pp. 90–100, July 1997.
[44] G. Miao, J. Zander, K. W. Sung, and S. B. Slimane, *Fundamentals of Mobile Data Networks*. Cambridge University Press, 2016.
[45] I. Adan and J. Resing, *Queueing theory*. Eindhoven University of Technology Eindhoven, 2002.
[46] E. Vanin, "Analytical model for optical wireless OFDM system with digital signal restoration," in *Proceedings of the 2012 IEEE GLOBECOM Workshops*, December 2012, pp. 1213–1218.
[47] C. A. Brackett, "Dense wavelength division multiplexing networks: principles and applications," *IEEE Journal on Selected Areas in Communications*, vol. 8, no. 6, pp. 948–964, August 1990.
[48] D. P. Tian, "A Review of Convergence Analysis of Particle Swarm Optimization," *International Journal of Grid and Distributed Computing*, vol. 6, no. 6, pp. 117–128, December 2013.
[49] M. J. Islam, X. Li, and Y. Mei, "A time-varying transfer function for balancing the exploration and exploitation ability of a binary PSO," *Applied Soft Computing*, vol. 59, pp. 182–196, October 2017.
[50] O. Olorunda and A. P. Engelbrecht, "Radio Environment Aware Computation Offloading with Multiple Mobile Edge Computing Servers," in *Proceedings of the 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, June 2017, pp. 1–5.

Tiago Gama Rodrigues (M'15) received his Bachelor Degree in Computer Science from the Federal University of Piaui, in Brazil, in 2014. He has worked as a researcher in 2013 at Bucknell University, U.S.A., and in 2014 in Tohoku University, Japan. He received his M.Sc. degree from Tohoku University in 2017 and he is currently pursuing his Ph.D. degree in the same institution. He has previously been awarded a scholarship by the Coordination for the Improvement of Higher Education Personnel from Brazil to study for one year in Bucknell University in 2013, and currently receives a scholarship from the Japanese Ministry of Education, Culture, Sports, Science and Technology to perform his studies in Japan. He was the recipient of the Best Presentation Award in the A3 Foresight Program 2016 Annual Workshop, the IEEE VTS Japan 2016 Young Researcher's Encouragement Award, and the 2017 Tohoku University Graduate School of Information Sciences Dean Award.

Katsuya Suto (M'16) received the B.Sc. degree in computer engineering from Iwate University, Morioka, Japan, in 2011, and the M.Sc. and Ph.D. degrees in information science from Tohoku University, Sendai, Japan, in 2013 and 2016, respectively. He is currently a Postdoctoral Fellow in the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada. His research interests include resilient networking, software-defined networking, and mobile cloud computing. He received the Best Paper Award at the IEEE 79th Vehicular Technology Conference in 2013, the IEICE Academic Encouragement Award in 2014, the IEEE VTS Japan 2015 Young Researcher's Encouragement Award, the Best Paper Award at the IEEE/CIC International Conference on Communications in China in 2015, and the Best Paper Award at the IEEE International Conference on Communications in 2016. He is a member of the IEICE.



Hiroki Nishiyama (SM'13) received the M.S. and Ph.D. degrees in information science from Tohoku University, Japan, in 2007 and 2008, respectively. He is an Associate Professor with the Graduate School of Information Sciences, Tohoku University. He has published over 160 peer-reviewed papers including many high-quality publications in prestigious IEEE journals and conferences. His research interests cover a wide range of areas including satellite communications, unmanned aircraft system networks, wireless and mobile networks, *ad hoc* and sensor networks, green networking, and network security. He was a recipient of the Best Paper Awards from many international conferences, including IEEE's flagship events, such as the IEEE Global Communications Conference (GLOBECOM) in 2014, 2013, and 2010; the IEEE International Conference on Communications (ICC) in 2016 (ICC); and the IEEE Wireless Communications and Networking Conference (WCNC) in 2014 and 2012, the Special Award of the 29th Advanced Technology Award for Creativity in 2015, the IEEE ComSoc AsiaPacific Board Outstanding Young Researcher Award 2013, the IEICE Communications Society Academic Encouragement Award 2011, and the 2009 FUNAI Foundations Research Incentive Award for Information Technology. He currently serves as an Associate Editor for Springer *Journal of Peer-to-Peer Networking and Applications*, and the Secretary of IEEE ComSoc Sendai Chapter. One of his outstanding achievements is Relay-by-Smartphone, which makes it possible to share information among many people by device-to-device direct communication. He is a Senior Member of the IEICE.

Nei Kato (F'13) is a Full Professor and the Director of Research Organization of Electrical Communication, Tohoku University, Japan. He has published over 350 papers in prestigious peer-reviewed journals and conferences. He has been engaged in research on computer networking, wireless mobile communications, satellite communications, *ad hoc* and sensor and mesh networks, smart grid, IoT, Big Data, and pattern recognition. He has been the Editor-in-Chief of the *IEEE Network Magazine* since 2015, the Associate Editor-in-Chief of the IEEE Internet of Things Journal since 2013, an Area Editor of the IEEE Transactions on Vehicular Technology since 2014, and the Chair of the IEEE Communications Society Sendai Chapter. He served as a Member-at-Large on the Board of Governors, IEEE Communications Society from 2014 to 2016, the Vice Chair of Fellow Committee of IEEE Computer Society in 2016, and a member of IEEE Computer Society Award Committee from 2015 to 2016 and IEEE Communications Society Award Committee from 2015 to 2017. He has also served as the Chair of Satellite and Space Communications Technical Committee from 2010 to 2012 and *Ad Hoc* and Sensor Networks Technical Committee of IEEE Communications Society from 2014 to 2015. He is a Distinguished Lecturer of IEEE Communications Society and Vehicular Technology Society. He is a fellow of the IEICE.

Katushiro Temma (M'10) received the B.E., M.E., and Ph.D. degrees in communications engineering from Tohoku University, Sendai, Japan, in 2010, 2012, and 2016, respectively. From April 2013 to March 2015, he was a Japan Society for the Promotion of Science (JSPS) research fellow. Since April 2016, he has been with the National Institute of Information and Communications Technology (NICT). He is a recipient of the IEEE VTS Japan 2011 Young Researcher's Encouragement Award. He is a member of the IEICE.