

# A Study on Intelligent Design and Control for Mobile Edge Computing Systems under the Wireless Environment

無線環境下におけるモバイルエッジコンピューティングシステムの  
知能設計と制御に関する研究

A dissertation presented  
by

Tiago Koketsu Rodrigues

submitted to  
Tohoku University  
in partial fulfillment of the requirements  
for the degree of

Doctor of Philosophy

Department of Applied Information Sciences  
Graduate School of Information Sciences  
Tohoku University

January, 2020



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Background . . . . .	1
1.2	Research Objective . . . . .	5
1.3	Summary and Organization of the Thesis . . . . .	7
<b>2</b>	<b>Mobile Edge Computing Model</b>	<b>9</b>
2.1	Introduction . . . . .	9
2.2	Service Model . . . . .	10
2.3	Challenges . . . . .	17
2.3.1	Offloading Decision . . . . .	17
2.3.2	Resource Allocation . . . . .	18
2.3.3	Server Deployment . . . . .	19
2.3.4	Overhead Management . . . . .	20
2.4	Mathematical Model . . . . .	21
2.4.1	Assumed Scenario . . . . .	21
2.4.2	User Mobility . . . . .	24
2.4.3	Number of Users . . . . .	25
2.4.4	Transmission Delay . . . . .	26
2.4.5	Processing Delay . . . . .	31
2.4.6	Backhaul Delay . . . . .	33
2.4.7	Service Delay . . . . .	34
2.5	Summary . . . . .	34
<b>3</b>	<b>Machine Learning-based Resource Allocation</b>	<b>38</b>
3.1	Introduction . . . . .	38

3.2	MEC Resource Allocation in the Literature . . . . .	39
3.3	Particle Swarm Optimization . . . . .	42
3.4	Performance Evaluation . . . . .	44
3.5	Summary . . . . .	47
<b>4</b>	<b>Intelligent Edge Server Deployment Policy</b>	<b>51</b>
4.1	Introduction . . . . .	51
4.2	Shortcomings of Current MEC Deployment Strategies . . . . .	52
4.3	Machine Learning-based Selection of Base Stations . . . . .	54
4.3.1	Particle Swarm Optimization . . . . .	55
4.3.2	k-Means Clustering . . . . .	56
4.3.3	Proposed Algorithm . . . . .	58
4.4	Performance Comparison of Different Policies . . . . .	60
4.5	Summary . . . . .	66
<b>5</b>	<b>Cloudlet Activation Method in Dynamic MEC</b>	<b>67</b>
5.1	Introduction . . . . .	67
5.2	History of Improving MEC Scalability . . . . .	69
5.3	Artificial Intelligence for Activating Cloudlets and Configuring MEC	74
5.4	Proposal Evaluation . . . . .	78
5.4.1	Assumed Scenario . . . . .	79
5.4.2	Hyper-Parameter Analysis . . . . .	80
5.4.3	Application Profile Analysis . . . . .	84
5.5	Summary . . . . .	89
<b>6</b>	<b>Conclusion</b>	<b>90</b>
	<b>Acknowledgments</b>	<b>96</b>
	<b>Bibliography</b>	<b>99</b>
	<b>Publications</b>	<b>118</b>
	<b>Appendix</b>	<b>123</b>

# List of Figures

2.1	Comparison between SaaS, PaaS, and IaaS, specially in what is offered by the users and what is offered by the Service Providers in each service model. © 2019 IEEE . . . . .	12
2.2	Diagram illustrating a generic view of the service model in MEC (although Access Point and Cloudlet are virtually different entities, they are usually physically together and always at the edge). © 2019 IEEE . . . . .	13
2.3	Illustrative example of six users and two cloudlets and their associations. © 2018 IEEE . . . . .	22
3.1	Illustration of Procedure 2 being executed. In the initial moment, $\alpha$ is overworked while $\beta$ and $\gamma$ have wasted resources. After executing the procedure, the green and blue VM servers have been migrated away from $\alpha$ to balance the workload, and $\alpha$ has lowered its transmission power level while $\beta$ and $\gamma$ raised theirs, to better reflect their associated users' positions. This way, all cloudlets can efficiently reach their users, and work is equal. © 2017 IEEE . . .	45
3.2	Results for Study Case 1. © 2017 IEEE . . . . .	48
3.3	Results for Study Case 2. © 2017 IEEE . . . . .	49
3.4	Results for Study Case 3. © 2017 IEEE . . . . .	49
4.1	How the proposed algorithm works by assigning users to cloudlets with kMC and then using PSO to adjust the transmission power levels. © 2019 IEEE . . . . .	58

## LIST OF FIGURES

---

4.2	MEC service delay variation as the number of base stations is increased. © 2019 IEEE . . . . .	63
4.3	MEC service delay variation as the number of cloudlets is increased. © 2019 IEEE . . . . .	64
4.4	MEC service delay variation as the number of users is increased. © 2019 IEEE . . . . .	65
5.1	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. © 2018 IEEE . . . . .	73
5.2	Maximum number of users served by our proposal with five cloudlets under different values for PSO local inertia ( $\vartheta$ ). © 2018 IEEE . . . . .	81
5.3	Maximum number of users served by our proposal with five cloudlets under different values for PSO local acceleration ( $\varrho_h$ ). © 2018 IEEE . . . . .	82
5.4	Maximum number of users served by our proposal with five cloudlets under different values for PSO global acceleration ( $\varrho_g$ ). © 2018 IEEE . . . . .	83
5.5	Probability that latency is below the threshold based on the number of users using Application A (no relevant difference between communication or computation burdens). © 2018 IEEE . . . . .	85
5.6	Probability that latency is below the threshold given the number of users using Application B (heavier burden on processing). © 2018 IEEE . . . . .	86
5.7	Probability that latency is below the threshold given the number of users using Application C (heavier burden on transmission). © 2018 IEEE . . . . .	87
5.8	Maximum number of users for all methods under all applications. © 2018 IEEE . . . . .	88

# List of Tables

1.1	Table summarizing the main characteristics of the different cloud computing models. . . . .	2
3.1	Study Cases Parameters . . . . .	47
4.1	Existing works in ECC server deployment and what is lacking on them. . . . .	54
4.2	Parameters used in the simulations. . . . .	62
5.1	Simulation Parameters . . . . .	79

# Chapter 1

## Introduction

### 1.1 Background

Cloud Computing [1, 2] is a service model where user devices are able to connect to cloud servers. The servers have a significantly wide pool of resources (such as communication bandwidth, energy reserves/sources, processors, computing capability, computer memory), a lot more than the user devices. Thus, the client devices can use those resources to execute more demanding jobs and tasks whose requirements go beyond what can be done locally by sending these tasks to the servers. The cloud denomination comes from the offloading to remote machines and the transparency of the service. Forbes [3] predicts that the cloud computing market will reach US\$411 billions by 2020, and many big companies already have established cloud services, such as Google [4], Amazon [5], and Microsoft [6].

Cloud computing can work with remote servers, located far away from the user device, which could be a static desktop computer in the case of Conventional Cloud Computing [1, 2] or a mobile device in the case of Mobile Cloud Computing [7, 8]. Mobile devices make the situation significantly different. Desktop computers sometimes can use cabled connections to reach the cloud servers,

Category	Server Location	Server Capacity	Number of Servers	Client Profile	Scenario Dynam-icity
Conventional Cloud Computing	Distant (backbone)	High	Few	Fixed	Low
Mobile Cloud Computing	Distant (backbone)	High	Few	Mobile	Medium
Edge Cloud Computing	Near (edge)	Low	Many	Fixed	Medium
Mobile Edge Computing	Near (edge)	Low	Many	Mobile	High

Table 1.1: Table summarizing the main characteristics of the different cloud computing models.

whereas mobile devices more often than not must depend on wireless and cellular networks in order to connect to the Internet and the cloud servers. Additionally, mobile devices are even more limited than desktop environments, with fewer resources and a smaller array of applications that can be executed locally [9]. This also means that they have more to benefit (i.e. the increase in the number of possible applications will be bigger) from cloud computing than desktop computers. However, the need to use wireless communication and even the device’s limitations complicate the deployment and the implementation of cloud computing services. To make matters more difficult, many mobile devices and mobile applications, such as Virtual / Augmented Reality programs [10], Tactile Internet applications [11], and Internet of Things (IoT) [12–14], need ultra-low latency, which cannot be possibly delivered by Mobile Cloud Computing due to the distance between the user devices and the cloud servers [15–18], which can sometimes be located in different continents [19]. To enable the execution of these applications, there have been recent efforts to create an Edge Cloud and

Mobile Edge Computing (MEC) [20,21]<sup>1</sup>, where the cloud servers are, instead of deployed remotely, installed on the edge of the network, near the users<sup>2</sup>. In order to make this possible, there must be an edge cloud server near all users, since the main point of MEC is a close-range connection between the mobile device and the server. However, this means a huge density of cloud edge servers, which incurs an incapacitating financial cost. To counterbalance this, these edge cloud servers are designed to be less powerful than the remote cloud servers, which are more concentrated but fewer in number. Due to this smaller capacity, they have been denominated cloudlets [27,28]. In summary, MEC offers lower latency and allows mobile devices to execute more demanding applications.

The possibility of using real-time applications and lifting device limitations has made MEC and cloudlets to be considered for future network designs along with the IoT [29–31] and 5G Networks [25]. However, there is another complicating issue in this caused by a key characteristic of IoT and 5G: a massive amount of devices. National Instruments estimates that 20 billions devices will be connected through 5G by 2020 [32], while Ericsson predicts 18 billion devices connected to IoT by 2022 [33]. Such scale cannot even be handled by Conventional Cloud Computing alone, stressing even further the need for MEC [14,34]. But, even with low quantities of devices, many problems related to MEC are too complex to be solved [35]. With this massive amount of devices also comes a

---

<sup>1</sup>Special note here to Fog Computing [22,23], a term defined by the OpenFog Consortium [24] which denotes, similarly to MEC, a paradigm where cloud computation is moved closer to the origin of the data and the requests, at the edge of the network. However, Fog Computing is usually utilized specifically for deploying servers in Local Area Networks gateways.

<sup>2</sup>It is important to say here that MEC is also used to denote Multi-Access Edge Computing, a term coined by the European Telecommunications Standards Institute [25, 26] to denote the usage of edge cloud servers by mobile networks as well as Wi-Fi and other fixed access technologies. Multi-Access Edge Computing is a more broad term, encompassing Edge Cloud Computing and Mobile Edge Computing, but we opted to go with mobile over multi-access to emphasize the dynamicity of the scenarios seen in the literature. Thus, MEC in this manuscript refers to Mobile Edge Computing. Nonetheless, many of the considerations found here can be applied to Multi-Access Edge Computing as well.

great variety of services and applications. Designing a MEC environment that takes into account the various kinds of wireless characteristics, device capabilities and service requirements is yet another layer of difficulty that cannot be solved by existing solutions and heuristical algorithms [36–38].

Table 1.1 summarizes the most important points between different types of cloud computing service. Note how the location of the server and the type of device vary between them. In this research project, we will focus on MEC and how to solve its issues of problem dimensionality, number of parameters and solution executability. Nonetheless, the considerations offered here could also be applied to the other service models due to the similarities between all of them.

One alternative for dealing with these issues of parameter dimensionality and algorithm feasibility in MEC is to use Machine Learning (ML) solutions [39, 40]. The base behind ML is to allow and enable the computer program to analyze and draw conclusions on the data by itself. The program is capable of doing this by learning the intricacies of the problem it is attempting to solve and making data-driven predictions and decisions, progressively improving its performance in one pre-determined task [41, 42]. In this area, the programmer’s job is to allow the program to learn the problem (i.e. train the program), which can be done by feeding it historic input and output, or, in other words, what is the output (performance)  $X$  of solution  $Y$  given the input  $Z$ . Given a large enough amount of tuples  $(X, Y, Z)$  and a method for evaluating the value of  $X$ , the program should be capable of drawing patterns and eventually creating efficient solutions [43, 44]. This definition is very generic and broad and there are many variations of it, but the fundamental is the same: ML programs are capable of finding solutions that could not normally be devised (or sometimes even understood) by its programmers. These solutions, albeit not necessarily optimal, are usually efficient enough. Moreover, and more important in this

discussion, current convex optimization<sup>3</sup> techniques applied to MEC rely on relaxation methods in order to deal with the high dimensionality of the realistic scenarios, i.e. they cannot be applied to the original problems [35, 41], which can only be tackled by ML [45]. Besides that, ML is also a very suitable option for dynamic scenarios, since ML models can find efficient solutions even with small changes in the problem, whereas conventional approaches may need a new execution.

## 1.2 Research Objective

In this thesis, the main goal is to develop a framework for intelligently configuring a MEC system. Our goal will be both to increase overall quality of service as well as allow service providers to achieve a higher profit. Thus, we will minimize the service latency experienced by users, which is key for a high quality service since it makes sure that cloud computing services are transparent and the offloading to the cloud is nearly unnoticeable (i.e. it feels as if all tasks are executed locally in the user device). Besides that, a lower service delay means that users do not have to be connected to the edge cloud servers for too long, freeing up the resources so they can be assigned to new users, consequently increasing the profit of the service providers. In addition to that, we will also configure the system so that the number of users that can be serviced concurrently, while respecting a delay threshold, is maximized while utilizing as few servers as possible, thus increasing money income and decreasing costs. Moreover, as mentioned before, MEC should be utilized by a massive amount of user devices as well as many servers and access

---

<sup>3</sup>ML itself is also used for optimization. Thus, to avoid confusion, we will use "convex optimization" when talking about the mathematical, more conventional variety of optimization and simply "optimization" when talking about learning-based techniques for optimizing a function.

points. Thus, all these solutions will be designed with scalability in mind and should have a low execution time even with many variables. The specific MEC problems we will tackle in this thesis are listed below:

- **MEC resource allocation to users**
- **Edge server deployment policy**
- **Server activation decision**

At first, we propose a method for allocating resources from MEC to the users. This means deciding which users are assigned to which edge cloud servers in a live scenario. This should be done in a way that no server is overloaded with too many users, as that would create queues that are too long for the resources of that particular server. A similar phenomenon can be observed with base stations, as too many users connected to the same base station would create high interference and collision around that base station. Thus, our method will balance user assignment in such a way that service delay in overall is minimized.

Secondly, we present an algorithm for deciding where each edge cloud server should be deployed. The location of the servers is significant as it affects the latency needed for users to access them. We assume that servers are always deployed in base stations, since this basically eliminates the transmission between the access point and the server and base stations themselves are usually positioned in strategic places already. However, it is not economically viable nor needed to put a server in every single base station, so a policy is needed to decide which base stations receive servers. Our proposal will do that while minimizing the overall service delay.

Finally, once again in a live scenario, we propose a method for deciding which servers should be turned on and which ones can be left turned off. This allows to effectively control how many resources are present in the system. The objective

here is to utilize as few servers as possible, to lower the cost incurred on service providers. Obviously, this should be done without compromising the quality of service given to users. Thus, we establish a service delay threshold that must be respected at all times. This also allows us to guarantee to users that their service will be completed within a certain, pre-determined time window.

### **1.3 Summary and Organization of the Thesis**

The remainder of thesis is organized as follows. Chapter 2 presents our detailed assumptions regarding the MEC system, including a mathematical model of the MEC with a formula for estimating the service delay. Chapter 3 shows our resource allocation method. Chapter 4 presents our server deployment policy method. Chapter 5 contains our server activation solution. Finally, concluding remarks are provided in Chapter 6.



# Chapter 2

## Mobile Edge Computing Model

### 2.1 Introduction

In this section, we will present the usual MEC service model that is generally utilized in the literature. This service model can be configured, where configuration is the set of options and parameters that operators can decide in order to deliver the best service possible following some pre-determined evaluation. To find these optimal parameters, some challenges and problems related to MEC have to be solved; such issues are also explored in this section, but not before a presentation of existing literature works on MEC.

The contents of this chapter refer to the following papers, which were written based on our own research.

- T. K. Rodrigues, K. Suto, H. Nishiyama, J. Liu and N. Kato, "Machine Learning meets Computation and Communication Control in Evolving Edge and Cloud: Challenges and Future Perspective," in *IEEE Communications Surveys & Tutorials*. Available online. © 2011 IEEE
- T. G. Rodrigues, K. Suto, H. Nishiyama, N. Kato and K. Temma, "Cloudlets Activation Scheme for Scalable Mobile Edge Computing with Transmission

Power Control and Virtual Machine Migration,” in *IEEE Transactions on Computers*, vol. 67, no. 9, pp. 1287-1300, September 2018. © 2011 IEEE

## 2.2 Service Model

As a prologue to the service model, it is important to explain the main entities in MEC [20, 27, 28, 46]. User Device (or user) is the one who contracts the service and will create the tasks (or input, or requests) that must be executed by the MEC system; additionally, the results (or output) of the tasks must be sent back to the user device after it is done. Access Point is the device that is used by the User Device to connect to the network and reach the MEC system, such as a base station with a cellular antenna. The User Device must utilize the Access Point to communicate with the other entities and the connection between Access Point and User Device is usually wireless. Cloudlet (or edge cloud server) is a physical computer that works as a server located at the edge of the network. The Cloudlet contains the resources needed for executing the requests created by the User Device. Compared to Conventional Cloud Servers, Cloudlets have lower capabilities (in computation, communication, etc.), hence their name. Virtual Machine (or virtual server) is a virtual environment located inside the Cloudlet, result of virtualization of the resources of the Cloudlet. This allows for the creation of multiple ”servers” even if there is only one single physical computer, making the separation of resources and their management easier. The Virtual Machine receives the requests by the User Device, executes the requests, and sends back the output. In order to execute these tasks, the physical server (e.g. Cloudlet) allocates some of its resources to the Virtual Machine according to the demands of the requests and a pre-determined policy. Each User Device is associated with a single Access Point, a single Cloudlet and a single Virtual Machine. Generally speaking, each Virtual Machine is only associated with one

User Device (thus, users cannot send their requests to other virtual machines, and the requests must be routed to the host of the virtual machine, regardless of where the user is and whether the user moved to somewhere since the creation of the virtual machine), while Access Points and Cloudlets can be associated with many User Devices. In physical terms, Cloudlets are usually located in the same location as Access Points, for convenience [27, 47–49]. Finally, there is also a Central Office which aggregates important information about the system (user location, cloudlet workload, etc.) and utilizes this knowledge to make decisions and configure the system.

Also of note is that in MEC (and cloud computing as a whole), there are different products that can be offered as a service. Those are divided between Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS) [50]. In SaaS, the whole application is offered by the cloud service provider, and the users just supply the data that serves as input and in return receive the corresponding output [51]. In PaaS, the service provider offers everything up to the operating system and the client provides the application that will be executed [52]. Thus, in PaaS, the user will send both its application and the input for it, the application will utilize the cloud resources, and then the user will collect the output. Finally, in IaaS the service provider makes available to the user its computers, storage, networking equipment through virtualized resources that the client can acquire as needed, while the user is responsible for deploying everything from the operating system onward [53]. In other words, the user provides and manages the platform, which will be run on top of the service provider’s resources. Note, however, that all these models operate with virtual servers on top of the physical resources from the provider, as shown in Figure 2.1. Thus, the structure based around the virtual server described previously can still be applied to MEC regardless of the product being offered.

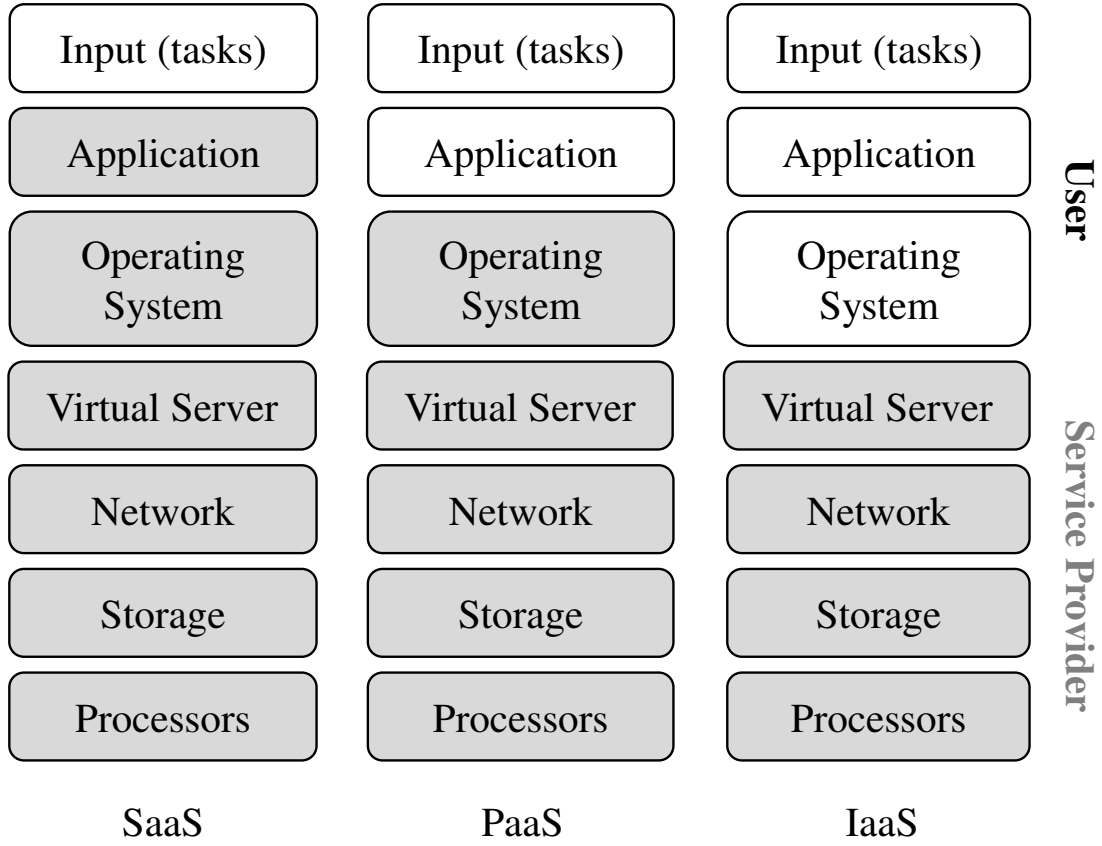


Figure 2.1: Comparison between SaaS, PaaS, and IaaS, specially in what is offered by the users and what is offered by the Service Providers in each service model. © 2019 IEEE

The MEC service model is usually assumed to be as follows [21, 27, 54–58]. When first entering the system, the user device makes an initial request that is received by the closest cloudlet. This request may be forwarded to a remote centralized entity (a central device with full information and control of the MEC system) or resolved in that edge cloud server. The result of this request is the creation of a Virtual Machine which will serve that user <sup>1</sup> (realistically speaking,

---

<sup>1</sup>The Virtual Machine is intrinsically connected to the service provided. Thus, if MEC is offering a specific application as a service, the Virtual Machine is capable of performing any routines related to that application. Conversely, if the platform or infrastructure is offered as

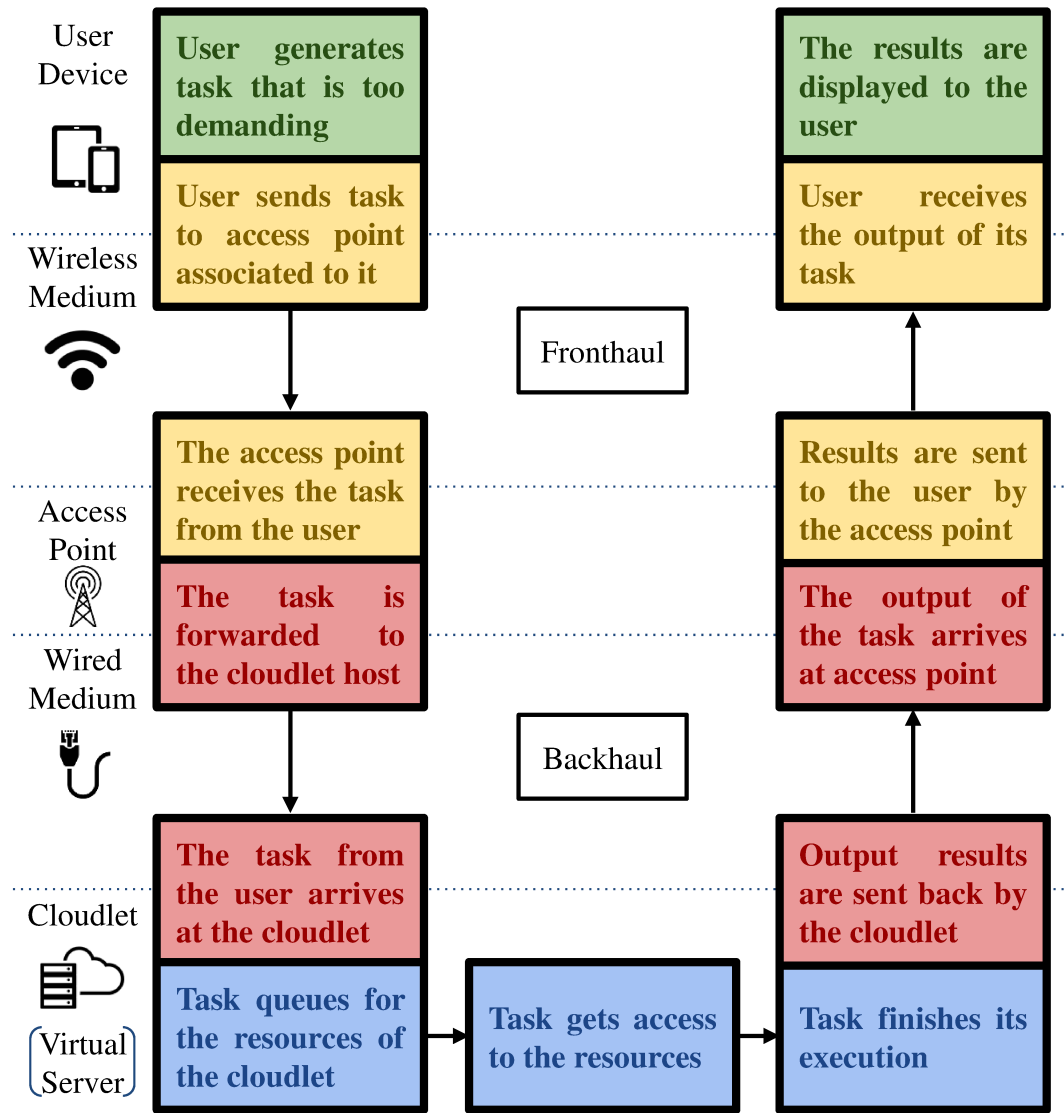


Figure 2.2: Diagram illustrating a generic view of the service model in MEC (although Access Point and Cloudlet are virtually different entities, they are usually physically together and always at the edge). © 2019 IEEE

there is an overhead related to virtualizing the resources of the host server when creating the virtual machine, which delays the beginning of the service by an

a service, the virtual machine can execute code sent by the user.

amount that varies depending on what exact service is offered [59]). As for what this service entails, it depends on the application, but it could go from task offloading and extending the environment of the user to even content delivery and caching, as mere examples. This virtual server will be hosted by one of the cloudlets of the system, with the choice of which one being made following a pre-determined policy. After this initial setup, the user will proceed to send tasks to its corresponding virtual server, who will receive this input and process them utilizing the local resources of the edge server. After finishing execution, the virtual server sends the output of the task back to the device, which displays it to the user as if it was a terminal. This communication between user device and edge cloud / virtual server happens wirelessly between the device and the access point and through a cable between the access point and the edge server. Figure 2.2 shows a diagram illustrating a typical MEC service model and all of its entities (note how the virtual server is inside the cloudlet). There are some relevant variations to this model, with the most prevalent being: multiple virtual servers [60,61], task fragmentation [62,63], virtual server migration [54,64], cooperation with remote servers [65,66], and virtual containers [67,68].

As a minor sidenote, one service model variation that deserves more attention is user mobility. The decision to consider user movement in the assumed scenario can indeed increase significantly the difficulty to solve a problem [69]. The issue comes from the uncertainty and dynamicity created by user mobility since the channel proprieties are always at risk of changing due to the user moving [49, 61, 70]. This movement could simply put more distance between the user and the access point, decreasing signal quality due to path loss, or even put the user in an area with more interference and obstacles. Consequently, it is very much a possibility that the user will transition to a state where a new access point is more favorable, which further complicates things with handover between the base

stations. Moreover, it may also mean that the user suddenly benefits more from connecting to a new edge server. Finally, all these changes are very difficult to predict, even if the users are assumed to follow some mobility models. Thus, a lot of MEC research prefers to ignore user mobility altogether [62, 65, 71], which is an option albeit not the best one in terms of choosing a realistic scenario.

As it is evident from the model and regardless of variations, service in MEC can be divided into two main segments: the transmission element (the communication between the user and the cloudlet) and processing element (the execution of the request sent by the user to the cloudlet) [21, 54, 55]. In the communication side, performance is mainly affected by interference between devices, bandwidth available, physical propagation, noise, transmission power and payload size [13, 62, 70, 72]. Meanwhile, in the computation side, performance mainly depends on server processing speed, size of the task (number of instructions, cycles required, etc.) and the competition for server resources [66, 73–75]. Some MEC applications will generate heavier communication payload (e.g. content delivery, where users may download videos and pictures that will put a lot of stress in the network), some MEC applications need to execute many instructions for their requests and will demand a lot of time from the processors at the cloud servers (e.g. image processing, where the images sent by the user devices must be fully analyzed while looking for points of interest), while some other MEC applications are a mix of both (e.g. big data analytics, where big databases must be transmitted and then analyzed at the processors). These show examples of high communication burden and high computation burden among MEC applications. These different profiles and requirements further complicate MEC operation, as the applications will need different resources and systems if an efficient service is desired (e.g. for content delivery, investment in bandwidth is desired, while for image processing, investment in more capable servers is desired; for servicing

both simultaneously, either the network is heavily and expensively equipped or the algorithm must be very efficient in resource allocation, which is obviously preferred) [76].

Additionally, it is worth noting that the service model in MEC is virtually identical to the one in Mobile Cloud Computing and Conventional Cloud Computing, involving Virtual Machines and job offloading. Consequently, many of the design and implementation problems in Mobile Cloud Computing and Conventional Cloud Computing are similar to what we find in MEC. However, there are some key differences. Firstly, due to lower latencies, MEC can service real-time applications and devices that demand quick response times on top of all applications and devices that conventional cloud computing handles, meaning that MEC has to work with a higher variety and bigger amounts of clients. Secondly, not only there are more clients, mobile devices themselves have features that cannot be ignored which may make local processing impossible or at least undesirable, such as their battery level, memory, and transmission data rate [31]. Those resources are particularly limited in mobile devices and their availability may change over time, which makes decisions on offloading, for example, even more complicated. Also, MEC also has to work with many more servers (actually, the concentrated servers in Mobile Cloud Computing can even be considered as one single giant server [77]). Finally, the networks in the edge are much more dynamic. These characteristics mean that problems in MEC have more parameters, higher dimensionality and incur higher volumes of data. For these reasons, solutions used in Mobile Cloud Computing / Conventional Cloud Computing often do not apply to MEC.

## 2.3 Challenges

With an objective function (or a combination of more than one [62, 78]) chosen, the cloudlet or the central office of the MEC service must then decide which configuration to utilize in order to optimize this objective. This configuration must answer possible questions regarding the service model of the system (e.g. which cloudlet will host the virtual server, which jobs will be offloaded). In the literature, this translates to a set of problems that are solved in order to raise the efficiency of the MEC system. These problems are usually independent of service variation or objective function. In the following sub-sections, we present categories of notable MEC problems.

### 2.3.1 Offloading Decision

Given one of the objectives previously mentioned, the class of problems related to offloading decision decides where the user-generated tasks should be executed. The choices can be local, inside the user device itself; the edge cloud servers, located near the user, following MEC; and the remote conventional cloud servers (as in Mobile Cloud Computing). In case of choosing the edge cloud servers or the remote cloud servers, a second possible choice is which server will execute the task, although this can be omitted in both cases (e.g. no virtual server migration so the host server is already decided; the remote cloud servers operate jointly as a massive super server [73, 77]). Moreover, the whole definition of the user task can vary between works, whereas some consider solid, indivisible tasks [73, 79] while others fragment them into subtasks (that can be dependent or independent between themselves) [62] and some others even duplicate tasks for redundancy [80]. Despite all these possible variations, the pattern is the same: decide where the user-generated task will be executed such that the objective function is optimized. Research work in this area usually concludes that offloading

decision (even in simplified scenarios with a single user or a single server) is either NP-hard [62] or NP-complete [73], such that relaxation or heuristic algorithms are usually the proposed solutions. However, such algorithms have their complexity order proportional to both the number of devices and servers, which renders them unfeasible in the future networks with massive amounts of agents.

### 2.3.2 Resource Allocation

Even if the offloading target is decided, service may be affected and changed by allocating more or fewer resources to the user. Because in real-life situations, both the user and the cloud server, either at the edge or at a remote location, have a limited amount of resources, the decision of how many to allocate to the MEC service is an important one in order to operate the network efficiently and to unleash the full potential of the system [65]. This resource allocation can be done either in the fronthaul, i.e. directly involving the user and its relation with the rest of the system, or in the backhaul, i.e. without direct relation to the user. The resource allocation decision can be made by a centralized office, with full [65,74] or partial knowledge [81] of the system, or individually by each server / user device [13,82]. These resources are mostly related to the communication side (sending input and output between the user device and the server), although there are some mentions to the applicability in the computation side (the execution of the task). For communication, examples are channel bandwidth [65, 74] and transmission power [54, 55], among others. All these affect the communication latency between the user and the server, so they consequently have effects on the service latency. Transmission power has a direct effect on the energy consumption of the nodes, while the communication delay itself has an indirect consequence in that the nodes may have to spend more time transmitting. The same can be said about profit, relating the money spent on energy and the monetary gain by

finishing more tasks in less time. Regarding resources in the computation side, notable examples are number of processors in a single server [65, 74] and even the speed of these processors [62], although research involving such parameters is rather scarce. Specifically related to backhaul resource allocation, notable mentions are servers sharing workload [72], servers sharing resources [83], virtual servers being migrated [61, 70], among others. The problem of deciding how many resources to allocate in order to optimize a system-wide metric is also usually solved by heuristic algorithms that are usually more complex if the model of the scenario is more realistic. Furthermore, since this allocation is taken with a system view, it is also affected by the number of devices and servers, which renders them unfeasible in the future networks with massive amounts of agents.

### 2.3.3 Server Deployment

A problem that is very characteristic of MEC is the choice of where the servers will be deployed. Being edge servers, there are usually multiple choices of where they could be installed. As mentioned before, they are typically located near network access points, but since it is not viable or necessary to have one server in each access point, there is still a choice to be made [47, 48]. The conclusion is that there must be a choice of where to deploy the servers, which naturally should be guided by an objective function, trying to optimize a pre-determined goal, such as the ones listed in our previous sub-section. This usually means deploying servers near places where users conglomerate, so more clients can be served (raising profit) with shorter connection distance [48, 84]. However, in MEC particularly, users can move, which may turn a deployment location from a desirable one into a bad position [49]. Moreover, differently from virtual machines, server deployments are usually permanent, or at the very least expensive to change, demanding extra care when choosing. Furthermore, the deployment choice also comes with important

financial consideration, as the number of servers and their locations all come with associated costs [85, 86]. The deployment places may have to be rented or equipment has to be acquired, making this an essential consideration even if the objective function is not based on the service provider's profit. From this discussion, it is clear that solutions to the deployment challenge depend heavily on how many servers will be deployed and how many choices are there, which in itself already means a complex scenario. Furthermore, the dynamicity of MEC means that the scenarios keep changing, which comes in contrast to the permanency of the deployment decision, making the choice of a good location even more difficult, especially in a future with the massive amount of servers and devices to be served.

### 2.3.4 Overhead Management

Finally, another point of note is how all these configurations and actions incur significant overhead. The choices of where to offload and how many resources to allocate are all performed online, meaning that until such decisions are made, the received requests will not be processed nor answered, increasing the latency and decreasing service quality [87, 88]. Moreover, there are other actions in the service model of MEC that result in overhead as well, particularly the ones related to the virtual server. Not only a choice has to be made of which physical server will host the virtual server for each client, the virtualization process itself and the initial setup of the virtual server takes time [59]. Additionally, in scenarios with virtual server migration, this transfer of the virtual machine between servers also takes time, during which the service is potentially paused [60]. Other possible causes for overhead cost include handover between networks when the user moves [89, 90], orchestrating the cooperation between different domains [19, 72], deciding the routing between the user device and the associated servers [91, 92], among others. Most works tend to ignore some if not all of these overhead sources,

but for a more realistic modeling of the scenario, they should be considered. However, not only the calculation of overhead itself is quite difficult, overhead costs themselves will probably ramp up as more devices and more servers are considered since these extra agents mean more complications for the algorithms responsible for calculating the configurations, which means they will take longer to execute [36,38].

## 2.4 Mathematical Model

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.

### 2.4.1 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

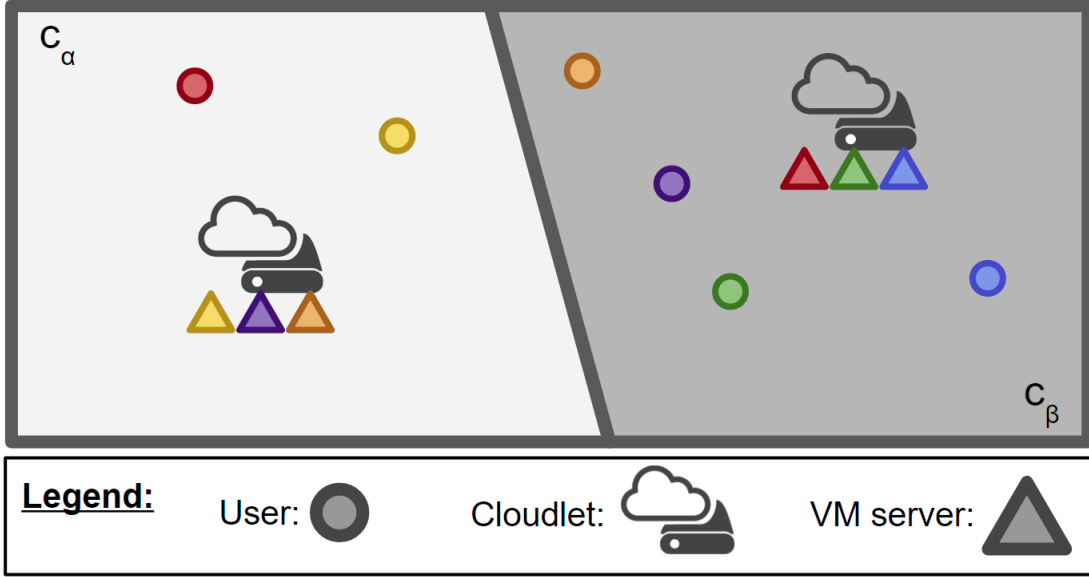


Figure 2.3: Illustrative example of six users and two cloudlets and their associations. © 2018 IEEE

under extreme stress conditions; if they behave well in this scenario, they are able to handle simpler cases as well [139]. There are  $|C|$  cloudlets in the scenario, which all belong to set  $C$ ; we denote them by  $c_i$ , with  $0 \leq i < |C|$ .

For now, we will assume that all base stations have a co-located cloudlet. This allows us to treat cloudlets both as servers and as base stations. Users have a physical and a virtual association [93]. 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 [94].

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  $\lambda_{c_i}(X, t_k) \leq \lambda(X, t_k)$ . Figure 2.3 illustrates the difference between those two values. Here,  $A_{c_\alpha}$  is light gray and  $A_{c_\beta}$  is dark gray. In the light gray area,  $\iint \lambda(X, t_k) dX$  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,  $\iint \lambda_{c_\alpha}(X, t_k) dX$  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  $c_\alpha$ . Analogously, in the dark gray area,  $\iint \lambda_{c_\alpha}(X, t_k) dX$  is two because two users in that area (the purple and orange ones) have their VM servers hosted by  $c_\alpha$ ; these users are not in the light gray area, i.e., not physically associated to  $c_\alpha$ , so they will use the backhaul

link and  $c_\beta$  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, transmission power levels can be changed to create a more efficient cloudlet configuration. Physical and virtual associations are then reset to the new resulting RSS Voronoi Diagram, which may lead to VMs being migrated.

For this mathematical model, 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.

### 2.4.2 User Mobility

We assume users move thusly [95]: 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  $\mathfrak{T}\mathbf{v}$  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).

$$E[\lambda(X, t_k)] = \alpha(X, t_k) + \iint_{\Omega(X)} \left( \frac{1}{2\pi} \zeta \left( \frac{r_X^Y}{\mathfrak{T}} \right) \lambda(Y, t_{k-1}) \right) dY \quad (2.1)$$

$$E[\lambda_{c_i}(X, t_k)] = \alpha_{c_i}(X, t_k) + \iint_{\Omega(X)} \left( \frac{1}{2\pi} \zeta \left( \frac{r_X^Y}{\mathfrak{T}} \right) \lambda_{c_i}(Y, t_{k-1}) \right) dY \quad (2.2)$$

### 2.4.3 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.

$$U_{A_{c_i}}(t_k) = \iint_{A_{c_i}} E[\lambda(X, t_k)] dX \quad (2.3)$$

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.

$$V_{c_i}(t_k) = \iint_{A_{c_i}} E[\lambda_{c_i}(X, t_k)] dX \quad (2.4)$$

#### 2.4.4 Transmission Delay

We assume that the channel capacity for transmission follows the Shannon Hartley theorem [96], 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 [97] spectral density (Watts per Hertz) and  $I$  is the sensed interference (Watts).

$$\mathfrak{C} = \mathfrak{B} \log_2 \left( 1 + \frac{S}{\mathfrak{B}N + I} \right) \quad (2.5)$$

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 [98]) and the path loss between

both nodes ( $L_{a_{RX}}^{a_{TX}}$ ), all given in decibels. It then converts this value to Watts.

$$S_{a_{RX}}^{a_{TX}} = 10^{(\omega_{a_{TX}} + G_{a_{TX}} + G_{a_{RX}} + H - L_{a_{RX}}^{a_{TX}})/10} / 1000 \quad (2.6)$$

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,  $r_{a_{RX}}^{a_{TX}}$  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.

$$L_{a_{RX}}^{a_{TX}} = L_1 + 10n_1 \log_{10} r_{a_{RX}}^{a_{TX}} + 10(n_2 - n_1) \left( 1 + \frac{r_{a_{RX}}^{a_{TX}}}{r_b} \right) \quad (2.7)$$

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  $\gamma$ ; 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

$$\dot{T}_{c_i}(t_k) = 2 \frac{g_{c_i}}{\gamma} + D_{c_i}^{\text{up}}(t_k) + D_{c_i}^{\text{down}} \quad (2.8)$$

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.

$$g_{c_i} = \frac{\iint_{A_{c_i}} (\lambda(X, 0) r_{c_i}^X) dX}{\iint_{A_{c_i}} \lambda(X, 0) dX} \quad (2.9)$$

In the uplink, we can calculate the average time needed to send a packet through (2.5) 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.

$$Q_{c_i}^{up} = \frac{\iint_{A_{c_i}} \left( \lambda(X, 0) \frac{p^{up}}{B^{up} \log_2 \left( 1 + \frac{S_{c_i}^X}{B^{up} N + I_{c_i}} \right)} \right) dX}{\iint_{A_{c_i}} \lambda(X, 0) dX} \quad (2.10)$$

For the uplink, we assume users of the same cloudlet follow round-robin scheduling [99] 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.

$$I_{c_i} = \sum_{\substack{c_j \neq c_i \\ c_j \in C}} \left( \frac{\iint_{A_{c_j}} (\lambda(X, 0) S_{c_i}^X) dX}{\iint_{A_{c_j}} \lambda(X, 0) dX} \right) \quad (2.11)$$

If the users follow a round-robin scheduling, then they follow an M/D/1 queue model [99, 100], where 1 represents the single channel for transmission, D means that the timeslot size is a constant value ( $\tau$ ), 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  $\Lambda_1$ ) 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 ( $\tau$ ) by the total time needed ( $Q_{c_i}^{\text{up}}$ ). 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

$$\phi_{c_i}(t_k) = \begin{cases} \Lambda_1 U_{c_i}(t_k) + \frac{1 - \frac{\tau}{Q_{c_i}^{\text{up}}}}{\tau}, & \text{if } Q_{c_i}^{\text{up}} > \tau \\ \Lambda_1 U_{c_i}(t_k) & , \text{otherwise.} \end{cases} \quad (2.12)$$

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

$$\varpi_{c_i}(t_k) = \frac{\phi_{c_i}(t_k) \tau^2}{2(1 - \phi_{c_i}(t_k) \tau)} \quad (2.13)$$

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  $Q_{c_i}^{\text{up}}$  and  $\tau$ . 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  $\tau$ , which leads us to the formula below.

$$D_{c_i}^{\text{up}}(t_k) = \left\lceil \frac{Q_{c_i}^{\text{up}}}{\tau} \right\rceil (\varpi_{c_i}(t_k) + \tau) \quad (2.14)$$

For the downlink, we assume that each user is allocated a fraction of the total bandwidth through OFDM [101]. 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  $B^{\text{down}}$  is the total downlink bandwidth (Hertz), the bandwidth allocated to a user in  $X_0$  is

$$\mathfrak{b}_{X_0}^{\text{down}} = \mathfrak{B}^{\text{down}} \frac{(S_{X_0}^{c_i})^{-1}}{\left( \iint_{A_{c_j}} (\lambda(X, 0) S_X^{c_i}) dX \right)^{-1}} \quad (2.15)$$

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.

$$I_{X_0} = \sum_{\substack{c_j \neq c_i \\ c_j \in C}} S_{X_0}^{c_j} \quad (2.16)$$

Finally, the average total time for transmission in the downlink is calculated in the same way as performed in (2.10), but using the average packet size for

downlink ( $p^{\text{down}}$ ), and the new values for bandwidth ((2.15)) and interference ((2.16)) instead.

$$D_{c_i}^{\text{down}} = \frac{\iint_{A_{c_i}} \left( \lambda(X, 0) \frac{p^{\text{down}}}{b_X^{\text{down}} \log_2 \left( 1 + \frac{S_X^{c_i}}{b_X^{\text{down}} N + I_X} \right)} \right) dX}{\iint_{A_{c_i}} \lambda(X, 0) dX} \quad (2.17)$$

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, multiplying 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.

$$T_{\text{delay}}(t_k) = \frac{\sum_{c_i \in C} \left( U_{A_{c_i}}(t_k) \dot{T}_{c_i}(t_k) \right)}{\sum_{c_i \in C} U_{A_{c_i}}(t_k)} \quad (2.18)$$

### 2.4.5 Processing Delay

We assume that the access to the processors at each cloudlet follows an M/M/k queue [100], 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  $\mu$ , 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 ( $\Lambda_1$ ) 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

$$\Lambda_{c_i}(t_k) = V_{c_i}(t_k) \Lambda_1 \quad (2.19)$$

Queuing theory tells us that the average occupation rate for cloudlet  $c_i$  at timeslot

$t_k$  can then be derived by

$$\rho_{c_i}(t_k) = \frac{\Lambda_{c_i}(t_k)}{k\mu} \quad (2.20)$$

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

$$\begin{aligned} \Psi_{c_i}(t_k) &= \frac{(k\rho_{c_i}(t_k))^k}{k!} \\ &\left( (1 - \rho_{c_i}(t_k)) \sum_{n=0}^{k-1} \frac{(k\rho_{c_i}(t_k))^n}{n!} + \frac{(k\rho_{c_i}(t_k))^k}{k!} \right)^{-1} \end{aligned} \quad (2.21)$$

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.

$$\dot{P}_{c_i}(t_k) = \Psi_{c_i}(t_k) \frac{1}{1 - \rho_{c_i}(t_k)} \frac{1}{k\mu} + \frac{1}{\mu} \quad (2.22)$$

Finally, the average Processing Delay for all users in the system can be obtained in the same way that (2.18) was calculated.

$$P_{\text{delay}}(t_k) = \frac{\sum_{c_i \in C} \left( V_{c_i}(t_k) \dot{P}_{c_i}(t_k) \right)}{\sum_{c_i \in C} V_{c_i}(t_k)} \quad (2.23)$$

### 2.4.6 Backhaul Delay

As mentioned before, all cloudlets are connected to a backhaul switch [94]. 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) [102], 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  $\mathfrak{R}_{c_i, c_j}$  (the order between  $c_i$  and  $c_j$  is irrelevant here), where, for all pairs of cloudlets in  $C$ ,  $\mathfrak{R}_{c_i, c_j}$  Hz are reserved exclusively for that pair. These  $\mathfrak{R}_{c_i, c_j}$  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  $t_k$ .

$$\mathbf{r}_{c_i, c_j}(t_k) = \frac{\mathfrak{R}_{c_i, c_j}}{\iint_{A_{c_i}} E[\lambda_{c_j}(X, t_k)] dX + \iint_{A_{c_j}} E[\lambda_{c_i}(X, t_k)] dX} \quad (2.24)$$

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.

$$B_{\text{delay}}(t_k) = \frac{\sum_{(c_i, c_j) \in C^2}^{c_i \neq c_j} \left( \iint_{A_{c_i}} E[\lambda_{c_j}(X, t_k)] dX \frac{p^{\text{up}} + p^{\text{down}}}{r_{c_i, c_j}(t_k)} \right)}{\sum_{c_i \in C} V_{c_i}(t_k)} \quad (2.25)$$

### 2.4.7 Service Delay

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

$$S_{\text{delay}}(t_k) = T_{\text{delay}}(t_k) + P_{\text{delay}}(t_k) + B_{\text{delay}}(t_k) \quad (2.26)$$

## 2.5 Summary

This chapter offers important considerations for modeling problems in MEC. This includes a basic service model, with its entities and variations. More importantly, this chapter presents broad categories of MEC challenges, which are a useful starting point for research. In this sense, it is important to also discuss what are the current directions and trends in each challenge. For example, in the Offloading Decision category, the final goal is obviously to utilize multiple and various servers, in the edge or not, as possible destinations [9, 27]. This means a lot of variables to consider, as discussed previously. Compounded by a high quantity of users, the resulting problem ends up being very difficult to solve. Thus, more and more literatures pieces published recently are using a user-centric approach, where they focus on solving the problem for a single user [82, 103, 104]. This means a simpler (and thus feasible) solution that could be applied in a user-by-user case. This idea is also applicable to and can be seen in resource allocation, both in the fronthaul and the backhaul [105, 106]. However, for resource allocation, it is also

notable how more and more research is being made combining both the Communication and Computation planes of the service model [81, 107]. Previously, a lot of MEC work would focus on a single aspect, e.g. how to minimize the transmission delay, how to maximize processor utilization [13]. However, both transmission and processing are important for the quality of service and they are also dependent on each other, so much so that modifying communication parameters will affect the computation part of MEC. Thus, it is necessary when building a realistic solution to consider the whole service model of MEC, despite how complex this is [76]. In the case of Server Deployment, it is interesting to investigate other deployment locations besides next to base stations. Obviously, this comes with the drawback of the connection between the edge server and the base station, to integrate it with the network, which must be considered in the final model and objective function. Nonetheless, there are some benefits to this, as there may be locations with lower monetary costs (e.g. cheaper rent) or better energy infrastructure to set up the edge servers, bringing benefits to the service provider [108]. Moreover, there can be already existing servers that perform different, local functions in the edge (e.g. servers in a company or university) that could secondarily be utilized as MEC servers [27]. These possibilities are not very touched in the literature. Finally, Overhead Management as a whole is not studied enough, so there is plenty of work that could be performed in this field yet [87, 88]. The estimation and calculation of the overhead cost associated with each action and the integration of this with the other decision (e.g. the offloading destination or resource allocation ones), including its impact in the objective function, are useful future perspectives. This goes especially in contrast with the many research works that mainly assume that their proposed solutions will either be performed offline always or will have negligible cost, which is not realistic in most scenarios [87, 88, 90]. Finally, the chapter concludes with a mathematical

representation of the assumed MEC model. This includes a method for estimating the service delay of the system, including the latency needed for processing the user-generated tasks and transmitting them from the users to the access points and from the access points to the cloudlets.



# Chapter 3

## Machine Learning-based Resource Allocation

### 3.1 Introduction

In this chapter, we propose a method for lowering both Processing Delay and Transmission Delay in a scenario with an undetermined amount of cloudlets. The proposal utilizes Virtual Machine Migration to move Virtual Machines related to users from one cloudlet to another, and Transmission Power Control at the base stations to determine the signal strength received at the users and thus the communication quality. Those main technologies are used for achieving the following goals: effectively lower Service Delay as much as possible, provide a high Quality of Service for various application profiles, and stay computationally feasible. We utilize a mathematical model with a Particle Swarm Optimization (PSO) model [109], a Machine Learning technique, to achieve a low execution time and high efficiency.

The contents of this chapter refer to the following papers, which were written based on our own research.

- 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. © 2011 IEEE
- 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)*, Paris, pp. 1-6, May 2017. © 2011 IEEE

## 3.2 MEC Resource Allocation in the Literature

As mentioned before, the main requirement in MEC, to lower the Service Delay, can be translated to lowering the Transmission Delay and the Processing Delay. We define Transmission Delay as the time required for the user to send its task to the cloudlet plus the time it takes for the cloudlet to send this task's output back to the user. Since these actions are performed in the wireless medium, it is straightforward that parameters related to the wireless environment are the ones who affect this delay. Meanwhile, we define Processing Delay as the time required, inside the cloudlet, for the task to be executed and its output to be produced. This involves the time the task spends in the processor queue, waiting for access to the processor, as well as the time the processor takes to actually execute the task. Therefore, this type of delay is intrinsically connected to the efficient utilization of the processors of the cloudlets. Finally, we define Service Delay as the simple sum of the Transmission Delay and the Processing Delay. Through this division, we can separate existing approaches to handle the Service Delay requirement in the literature into two classifications: those related to Processing Delay, and those

related to Transmission Delay.

In the computation side, existing methods for lowering Processing Delay are mainly based on balancing the workload between the physical cloudlet servers, attempting to avoid overwork and/or wasted resources in any of the cloudlets. Since such goal leads to efficient use of the processor resources, it is an effective way of controlling Processing Delay. This can be achieved by always associating new users with cloudlets that currently have the least amount of work, in a technique called Work Scheduling [110, 111]; this would balance the workload and guarantee that the difference in amount of hosted Virtual Machine servers between cloudlets is minimal. Another existing approach, called Virtual Machine Migration [112, 113], deals with fixing situations where the workload is already unbalanced; it is based on migrating Virtual Machine servers from busy cloudlets to underworked ones, avoiding the wasting of resources on the latter group and keeping the workload more balanced. Finally, one other possibility is to divide the task into mutually independent components that can be executed in parallel; some of these components are executed locally, in the mobile device, while others are offloaded to one or more cloudlets. This technique, called Application Partitioning [114–117], is based on deciding which components to offload and which cloudlets should be their destination, with the choice being based on minimizing Processing Delay by avoiding sending requests to servers that are already overworked and would take longer to execute the task than the mobile device, and also spreading your components so that no single server is overwhelmed.

As it was mentioned before, Transmission Delay is closely connected to the wireless medium where the transmission is performed. For this reason, approaches focused on communication elements mainly concentrate on metrics such as Signal to Interference plus Noise Ratio (SINR), latency, and throughput. One existing technique is called Access Point Scheduling [118, 119], where users are associated

to the cloudlet closest to them, in order to minimize transmission distance, which in turn improves signal quality and propagation time; some variations of this technique involve mixed offloading to cloudlets and conventional cloud servers [120]. Many parallels can be drawn between Access Point Scheduling and Work Scheduling, but the main difference that separates them is their ultimate goal: the former focuses on improving Transmission Delay, while the latter is concentrated on Processing Delay. Another communication based strategy is Transmission Power Control [121,122], where the transmission power of the cloudlets is carefully set with the final goal of lowering the latency. Due to its tight connection to signal quality, interference, and channel capacity [96], controlling transmission power is an efficient method of controlling Transmission Delay. Yet another approach related to communication elements is Resource Allocation [123,124], where the available resources, such as channel bandwidth, are thoughtfully allocated in order to provide fairness among users. This allows for provisioning a good quality service even to users in disfavorable contexts (such as locations with high levels of interference), which consequently results into an improved average Transmission Delay.

It can be seen that there are plenty of approaches for lowering Service Delay in MEC, both in the communication side and in the computation side. However, none of them attempts to integrate and combine elements of both sides, i.e. the approaches have had a single focus on either communication or computation instead of a dual focus. Our proposal, therefore, aims at lowering both Transmission Delay and Processing Delay. The strategy chosen for this is to utilize Transmission Power Control to manage the Transmission Delay, while making use of Virtual Machine Migration to lower Processing Delay. This would enable us to alter both types of delay, which is necessary to achieve our objectives of truly lowering Service Delay, and providing a high quality service independent of

the characteristics (e.g. heavier burden on communication, or a higher necessity of computation) of the application. This last part is important because more and more profiles of application are being utilized; for example, while scientific experiments usually involve time-consuming computations in the cloudlet and small input and output packets for transmission, database driven applications have short computation in the form of looking up an entry and long transmission to send the possibly great amount data that composes such entry. This means that for some users the Processing Delay will have a bigger effect, while for others the Transmission Delay will have a bigger effect. A method for lowering Service Delay must be able to handle all types of application however. Moreover, while there may be other delays involved in the Service Delay (e.g., delay to migrate the Virtual Machine servers, delay to start the service), in this paper we consider a long timescale, an infinity horizon, where other delays are irrelevant when compared to Processing Delay and Transmission Delay (since the former are related to actions performed much less frequently than the ones related to the latter).

### 3.3 Particle Swarm Optimization

Given the mathematical model of Service Delay from Chapter 2, it is intuitive to mathematically optimize it by making the transmission power levels the decision variables (since they decide user association and received signal strength, they relate to both Processing Delay and Transmission Delay) and minimizing Equation (2.26). This approach was taken in the literature before [54], using integrals and partial derivatives. The problem is that this is only feasible for small amounts of cloudlets (such as 2 in the reference), since that number corresponds to the number of transmission power levels and consequently decision variables. Moreover, brute force has a complexity that is exponential on the amount of decision variables, Linear Programming techniques do not apply since this is not a

linear equation system, and derivatives and integrals only work with few decision variables (anything else is too complex).

This is why we utilize PSO for optimizing Equation (2.26) instead. PSO [109] works with a set of particles that intelligently move in the search space (where each position is a possible solution), trying to improve the best local (i.e. for that particle) and global (i.e. among all particles) solutions. The intelligence part comes from the bias on movement, that tends to go towards the best solutions found so far. Through this mechanism, PSO is capable of nearly optimizing the fitness function at a low execution time [125]. For our PSO model, shown in Algorithm 1, the solutions will be configurations for the transmission power levels of all cloudlets; therefore, our search space is  $N$ -dimensional. Initial positions and speed for all particles are random. The fitness function,  $f(\cdot)$ , is Equation (2.26).  $R$  is the number of particles;  $L$  is the number of iterations;  $q_r$ ,  $v_r$  and  $h_r$  are respectively the position, the speed and the best local solution of particle  $r$ ;  $g$  is the best global solution;  $\vartheta$  is the inertia constant; and  $\varrho_h$  and  $\varrho_g$  are the acceleration biases for the best personal and global solutions respectively.

In our proposal, a central office (CO), which aggregates information about users and cloudlets and can control the servers, would execute Algorithm 2. The CO would use as input the topography of the scenario, execute the PSO algorithm (which means solving the equation model from Chapter 2) and, through this, find a transmission power configuration for all cloudlets that lowers the Service Delay. Figure 3.1 illustrates this process. The frequency of execution of the procedure depends on the scenario; more dynamic cases should execute the algorithm more often, since the overhead would be compensated by the corrections to the Service Delay, while more static cases can execute the algorithm less often.

---

**Algorithm 1** PSO model for MEC multiple cloudlets scenario
 

---

```

1: for all  $r \in R$  do initialize  $q_r$  as a random  $N$ -tuple
2: end for
3: for all  $r \in R$  do initialize  $v_r$  as a random  $N$ -tuple
4: end for
5: while executed iterations  $< L$  do
6:   for all  $r \in R$  do
7:      $y \leftarrow f(q_r)$ 
8:     if  $y < f(h_r)$  then  $h_r \leftarrow q_r$ 
9:     end if
10:    if  $y < f(g)$  then  $g \leftarrow q_r$ 
11:    end if
12:     $\varphi_h \leftarrow \text{randomInt}(0, 1)$ 
13:     $\varphi_g \leftarrow \text{randomInt}(0, 1)$ 
14:     $v_r \leftarrow \vartheta \cdot v_r + \varphi_h \cdot \varrho_h \cdot (q_r - h_r) + \varphi_g \cdot \varrho_g \cdot (q_r - g)$ 
15:     $q_r \leftarrow q_r + v_r$ 
16:  end for
17: end while
18: return  $g$ 
    
```

---



---

**Algorithm 2** Integrated transmission power and Virtual Machine migration control for Service Delay minimization
 

---

```

1: Collect the physical location of users and cloudlets
2: Use Algorithm 1 to find configuration for lowering Equation (2.26)
3: Set transmission power levels of cloudlets according to the configuration found
4: Decide user association based on Equation (2.6)
5: Execute Virtual Machine Migration if user association changed
    
```

---

### 3.4 Performance Evaluation

To evaluate the performance of our proposal, we prepared three study cases, with parameters shown in Table I. Each study case was run 100 times, with different randomly generated topologies (i.e. physical location of users and cloudlets) each run. Results shown here are the average across all 100 runs, calculated through the model in Section III. We assume users are static, and bandwidth and

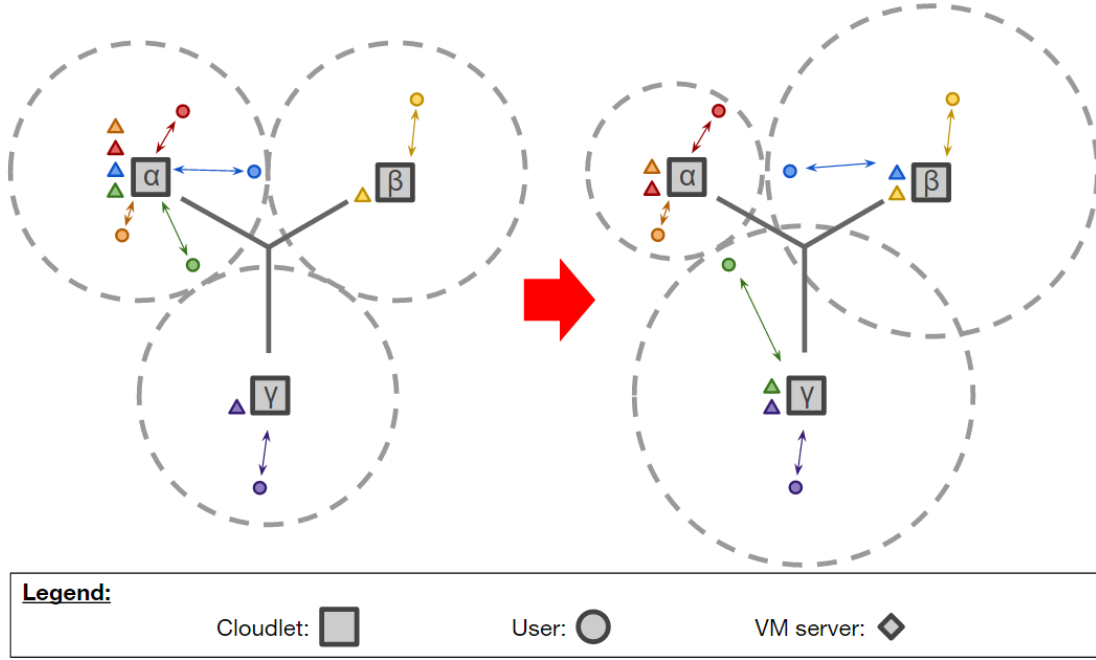


Figure 3.1: Illustration of Procedure 2 being executed. In the initial moment,  $\alpha$  is overworked while  $\beta$  and  $\gamma$  have wasted resources. After executing the procedure, the green and blue VM servers have been migrated away from  $\alpha$  to balance the workload, and  $\alpha$  has lowered its transmission power level while  $\beta$  and  $\gamma$  raised theirs, to better reflect their associated users' positions. This way, all cloudlets can efficiently reach their users, and work is equal. © 2017 IEEE

average packet size are the same both in uplink and downlink. For the path loss model, the level at 1m is 20dBm, the coefficients for short and long distances are respectively 2 and 4, and the breakpoint is 100m [126]. For the PSO model, there are 30 particles, the inertia constant is 0.7, and both accelerations are 2 [109].

For Study Case 1, we compared the performance of our proposal at each iteration with optimal values calculated through brute force. As seen in Figure 3.2, the Proposed Approach reaches values close to optimality, with the difference being under 5ms after 50 iterations. However, the proposal manages this with an execution time 11.57% that of the method used for calculating the optimal value;

and, as mentioned before, this value tends to be smaller in a higher scale. Thus, Study Case 1 shows how the Proposed Approach is a valid and still computationally feasible way of approaching optimality.

For Study Cases 2 and 3, we compared the Proposed Approach with two conventional ones. For the No Migration Approach, users send their tasks to the closest cloudlet, which also hosts their corresponding VM servers; this results in minimum transmission distance, but can lead to congestion, as too many users try to send packets to the same cloudlet, and overwork, as a single cloudlet hosts too many VM servers. In the Conventional Approach [112], users also send their tasks to the closest cloudlet, but now we assume VM servers are migrated as to result in all cloudlets hosting the same amount of VM servers; this gives minimum Processing Delay, but can still lead to congestion (tasks sent to cloudlets which do not host the corresponding VM server are transmitted through a wired connection of insignificant latency to the correct cloudlet, with the task output doing the opposite route afterwards). Regarding complexity, both conventional approaches and the proposal are  $O(N \cdot M)$ , where  $N$  and  $M$  are respectively the number of cloudlets and users. In Study Case 2, computation burden is varied in the form of the average task execution time, and in Study Case 3, communication burden is varied in the form of packet size. The results for Study Case 2 (Figure 3.3) and Study Case 3 (Figure 3.4) show how the No Migration Approach and the Conventional Approach have similar performances, while the proposal is consistently better. The difference to the Conventional Approach is smaller when computation use is high, since this approach minimizes Processing Delay, but the proposal has an advantage because it is the only one to consider Transmission Delay. This is more evident as the use of communication increases. The results show how a dual focus approach leads to significant improvement in performance when compared to single focus. It also shows how the proposal can deal with var-

Table 3.1: Study Cases Parameters

Study Case	1	2	3
Average packet size	500kB	500kB	0.5 to 1.5MB
Average task service time	500ms	50 to 1500ms	500ms
Number of cloudlets	3	40	
Number of users	120	500	
Number of PSO iterations	250	100	
Total area size	0.04km <sup>2</sup>	0.25km <sup>2</sup>	
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 <sup>-19</sup> W/Hz		
Wireless propagation speed	3*10 <sup>8</sup> m/s		
Processors per cloudlet	8		
Single user task arrival rate	6 tasks/min		
Round robin timeslot	85ms		

ious application profiles (i.e. mix of computation and communication burdens) better than the conventional approaches. The improvement (of between 0.2s and 1.3s) may seem small for a single task, but users' jobs are composed of multiple tasks, increasing the importance of the proposal's performance enhancement.

### 3.5 Summary

In this chapter, we proposed a PSO-enhanced method of lowering Service Delay in MEC together with a mathematical model for calculating Service Delay. The proposed method, which considers both communication and computation

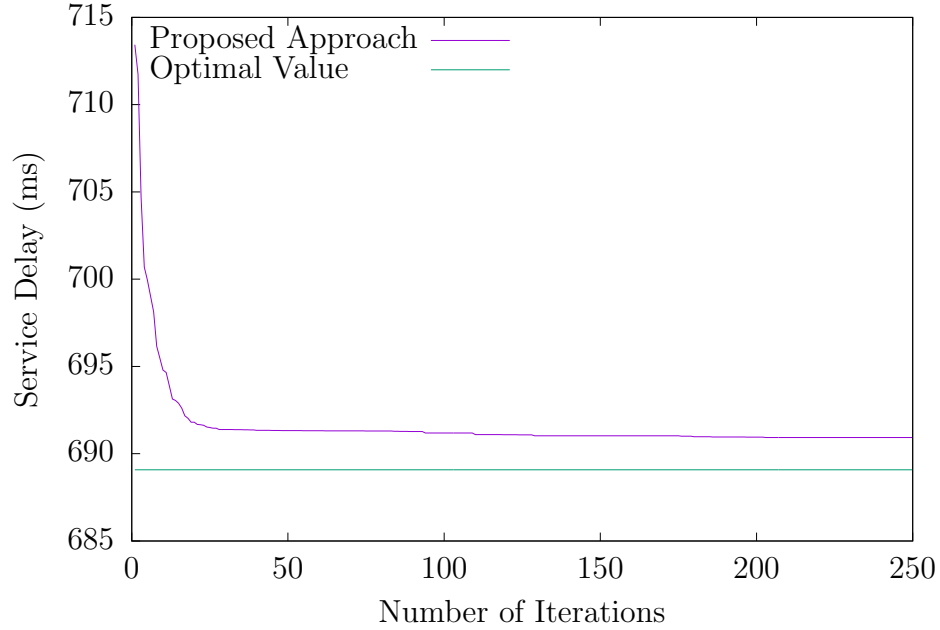


Figure 3.2: Results for Study Case 1. © 2017 IEEE

elements, was shown to be significantly better than an existing approach which only improves Processing Delay. This corroborates our theory that a combined method of improving Transmission Delay and Processing Delay is the most efficient way of dealing with Service Delay. The proposal was also shown to be near optimal while being computationally feasible. Since the expectation in the next generation of mobile devices is to rely more on communication (due to higher data rates), our proposal will be even more relevant, since it was shown to be superior specially in scenarios with high transmission burdens.

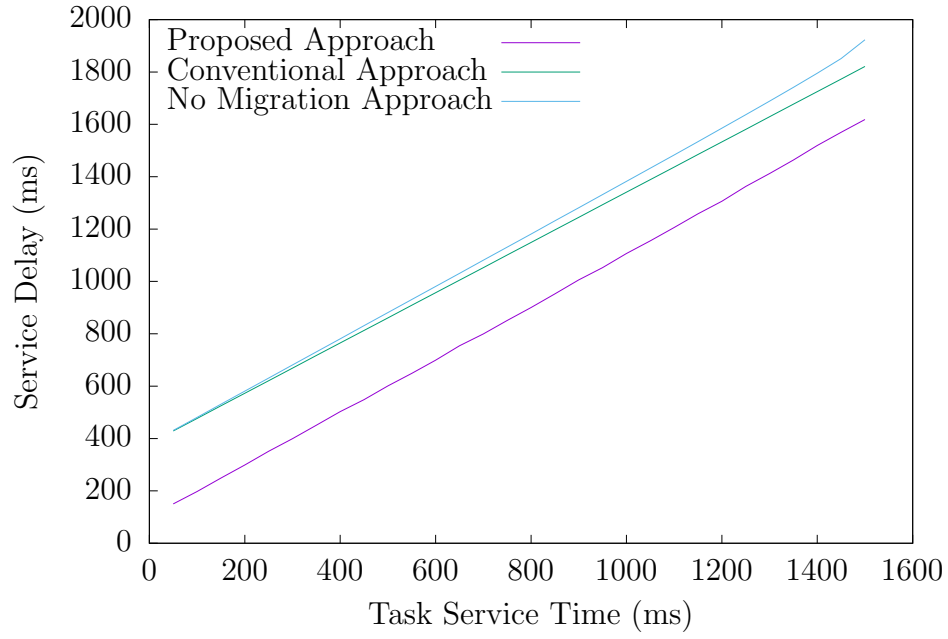


Figure 3.3: Results for Study Case 2. © 2017 IEEE

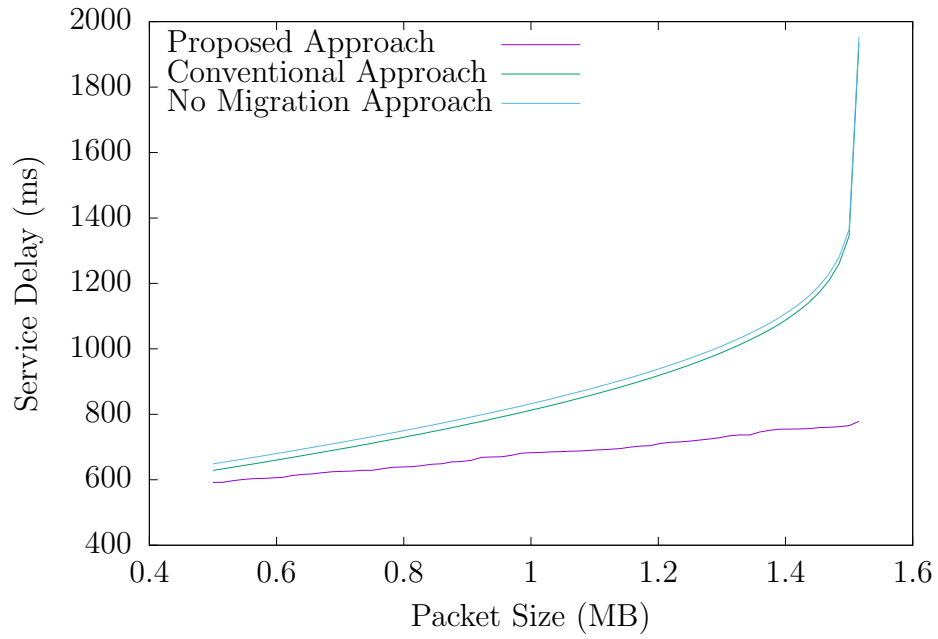


Figure 3.4: Results for Study Case 3. © 2017 IEEE



# Chapter 4

## Intelligent Edge Server Deployment Policy

### 4.1 Introduction

Since cloudlets are not all located in a single physical place as in conventional cloud computing, we must actively decide where to deploy them. Most works consider that cloudlets are deployed in base stations [48, 49, 55], and for good reasons. Base stations are often in central positions for reaching users and having cloudlets co-located with them also further lowers the latency as the propagation between the base station and cloudlet becomes negligibly low. Nonetheless, it is too expensive and usually not necessary to put a cloudlet in each base station, so there is still a decision to be made regarding which base stations will receive cloudlets.

In this chapter, we propose a cloudlet deployment policy for MEC based on the Machine Learning algorithms k-Means Clustering (kMC) and Particle Swarm Optimization (PSO). The objective of our proposal is to find a deployment solution where cloudlet resources are used as efficiently as possible while service

delay is minimized. Efficient resource usage means lower operational costs for service providers and lower service price for clients while a low service delay is essential for real-time and latency-intolerant applications (as well as maintaining transparency in cloud systems in general). Furthermore, the use of Machine Learning should allow our solution to be applicable in 6G IoT environments, i.e. scenarios with massive amounts of users and frequent requests. Our proposal will decide simultaneously where the cloudlets should be deployed and how to allocate their resources among the clients of the service. To prove the efficacy of our idea, the proposal's performance is compared with other deployment policies under a scenario that mimics 6G IoT.

The contents of this chapter refer to the following paper, which was written based on our own research.

- T. K. Rodrigues, K. Suto and N. Kato, "Edge Cloud Server Deployment with Transmission Power Control through Machine Learning for 6G Internet of Things," *IEEE Transactions on Emerging Topics in Computing*. Available online. © 2011 IEEE

## 4.2 Shortcomings of Current MEC Deployment Strategies

Intuition tells us that deciding the location of the cloudlets based on a pre-determined objective function would result in better service. Despite this, the vast majority of MEC research works with the assumption that servers are in already defined and immutable positions. Since no deployment policy is explicitly mentioned, it is safe to say that the cloudlets are randomly placed in the service area. Some research works directly state so [54, 55]. However, ignoring cloudlet placement decision means the final service is not optimized, even if the cloudlets

themselves are configured in a way that resource allocation is optimized. Thus, to offer the best service possible (in our case, the one with the lowest delay), we will carefully decide where to deploy the cloudlets.

Some few research works agree with us and thus proposed cloudlet deployment policies of their own. K. Xiao *et al.* [86] utilize a Markov chain structure to model the whole MEC service, including both communication and computation phases. This model is then used to predict workload for each cloudlet in each possible position and then decide deployment so that there is no overload (i.e. work is balanced across all servers). However, their solution is not exactly scalable and would result in extremely long execution times in scenarios with many servers and, more worryingly, base stations. Thus rendering their proposal unfeasible for IoT and 6G.

Some authors offer more scalable proposals for the cloudlet deployment problem. Y. Li and S. Wang [85] use Machine Learning in the shape of PSO to decide where to deploy the cloudlets. They model the energy consumption of the whole system and then use PSO to find out the locations that lead to higher energy efficiency. Alternatively, B. Li *et al.* [84] use kMC instead as their Machine Learning algorithm. For this research work, the goal was to minimize service delay and kMC was utilized to choose cloudlet locations that would lead to the lowest completion times for the users. Both works utilize Machine Learning and thus provide solutions that can handle high numbers of clients, servers and requests. However, the service model considered by them is lacking. They properly model the computation aspect of MEC, but in the communication side, they ignore collision and contention of resources, only considering a constant latency. Consequently, if applied to real-world scenarios, their solutions would not lead to optimal results as important aspects which they ignored would inevitably arise.

Table 4.1 summarizes the related works in MEC deployment and what is

Table 4.1: Existing works in ECC server deployment and what is lacking on them.

Literature reference	Shortcoming
Random deployment (most of the literature)	No objective function to guide deployment means that the final performance is usually not optimal
K. Xiao <i>et al.</i> [86]	Complexity depends on number of servers and base stations making it not feasible for 6G and IoT
Y. Li and S. Wang [85], B. Li <i>et al.</i> [84]	Ignores communication element of ECC, resulting in an overall less than optimal performance

missing in each of them. In this chapter, we propose a MEC cloudlet deployment solution that is both feasible and which considers the entire service model. Feasibility even in scenarios with many servers and/or clients will be achieved by using Machine Learning in the shape of kMC and PSO. Meanwhile, our service model will consider both computation and communication resources and their allocation, which is essential to realistically represent MEC [76].

### 4.3 Machine Learning-based Selection of Base Stations

From Chapter 2, we can see that the Service Delay is highly controlled by the associations of the users, both the virtual and physical ones. Logically, a proper solution would contain mechanisms for choosing such associations carefully to minimize the latency. Additionally, as mentioned before, users physically associated with the base station that offers the highest signal power. As shown by Equation (2.6), the easiest way to do this would be to utilize Transmission Power Level Control [54] on the base stations to individually choose their transmission

power levels and through that control how many and which users connect to each one. We can directly assign the virtual associations, i.e. which cloudlet serves which user. For the transmission power level configuration, we will use PSO as specified in [55]. In that work, the machine learning algorithm is used for balancing the communication workload between the base stations. However, for that work, transmission power is used to decide both physical and virtual associations. This results in a simple solution, but has the drawback of not allowing optimal configuration: if the optimal solution has users physically associated with a base station and virtually associated to a cloudlet co-located with a different base station, the solution in [55] will never find it. Indeed, that solution ignores these configurations completely. Thus, we will adapt PSO to only configure the transmission power levels and thus only choose physical associations. For the virtual association, we will utilize kMC.

### 4.3.1 Particle Swarm Optimization

PSO [127] works by having multiple agents looking on the space of all possible solutions for the optimal configuration. Each position in this space represents a possible solution to the problem being tackled by PSO. The agents, called particles, move around this space, evaluating each the solutions corresponding to the positions they stop on before moving to a new position. Their movement is biased towards the best solution found so far globally, i.e. they swarm around the best solution found by the set of all particles. To avoid getting stuck in local optimum points, the movement is also biased towards what is called local best, which is the best solution found by that particular particle and biased towards the direction and speed of their previous movement, which is called the inertia component. These three elements (global best, local best, and inertia) have weights associated with them to dictate how much they influence the solution

search. Moreover, the swarming behavior around the global/local best solutions is usually also conditioned by random variables that can take any value between 0 and 1 that change in each iteration. The goal of these elements is to balance the search between exploitation (the swarm around the best solutions found) and exploration (the random search for new elements to avoid getting stuck in local optima).

In our problem, we want to find the optimal configuration of transmission power levels for all base stations. Thus, our solutions are tuples of  $\mathfrak{V}$  elements, one transmission power level for each base station. We will assume that virtual associated are already defined before the algorithm starts, so PSO can only change physical associations. The objective function that will dictate what the global/local best are will be Equation (2.26) so our algorithm can find the configuration that leads to minimum delay for our users. Our implementation of PSO is shown in Algorithm 3.  $\mathbb{R}^{\text{PSO}}$  is the total number of PSO iterations we want to run. The particles are represented by  $\mathbb{P}$ .  $q_{\mathbb{P}}$  and  $\mathbb{V}_{\mathbb{P}}$  are correspondingly the position and velocity of particle  $\mathbb{P}$ .  $\mathbb{B}_{\mathbb{P}}^l$  is the best local solution found by particle  $\mathbb{P}$  so far while  $\mathbb{B}^g$  is the global best solution found by all particles so far.  $\mathbb{F}^l$  and  $\mathbb{F}^g$  are the random variables applied respectively to the local and global best that change with each iteration.  $\mathbb{W}^i$  is the inertia weight,  $\mathbb{W}^l$  is the local best weight and  $\mathbb{W}^i$  is the global best weight. Finally,  $f(\cdot)$  represents our objective function.

### 4.3.2 k-Means Clustering

kMC [128] is an algorithm for dividing elements into clusters based on distance. The total number of clusters is decided beforehand to be  $K$ . The algorithm starts by selecting a random location for the centroid of each one of its  $K$  clusters. Then, for each element, it will be associated with the centroid that is physically closest to them. After this phase, for each cluster, the centroid is moved to the

average center of all elements of that cluster. Then, all elements are assigned new clusters based on these new centroids. This cycle of assigning elements to clusters/moving the centroids based on the elements is repeated until no centroid changes location. The overall performance of the algorithm is dependent on the initial random centroids. To eliminate this bias, this entire process is repeated for multiple iterations, with different initial random locations for the centroids. For each iteration, the performance of the final clusters, as judged by an objective function, is recorded. The output of kMC will be the clusters from the iteration that resulted in the best performance.

For our problem, we will make some adaptations to kMC. We will use kMC for deciding the virtual association of the users. Thus, we will use the algorithm for deciding  $K$  clusters, one for each cloudlet. As such, the location of the cluster centroids (i.e. cloudlets) will be limited to the location of the base stations. We will assume that physical associations are already decided based on the best offered signal power at each user before the algorithm starts and that kMC will only change the virtual associations. Consequently, using physical distance to determine which cluster the users will join is not the best choice. Instead, users will cluster with cloudlets whose co-located base station has the shortest backhaul path in terms of propagation to the physical association of the corresponding user. Only in case of ties, we will utilize the physical distance between user and cloudlet. Moreover, as is, the algorithm is susceptible to overloading cloudlets by creating clusters that are too big in cases where many users are concentrated around a base station. This is bad because it leads to long processing delays. Thus, we will limit the cluster sizes so that all cluster have the same number of users (in our case, that would be  $\mathcal{U}/\mathcal{K}$ ). Finally, the iteration with the best configuration of clusters will be determined through Equation (2.26). Our implementation of kMC will be shown in the next subsection.

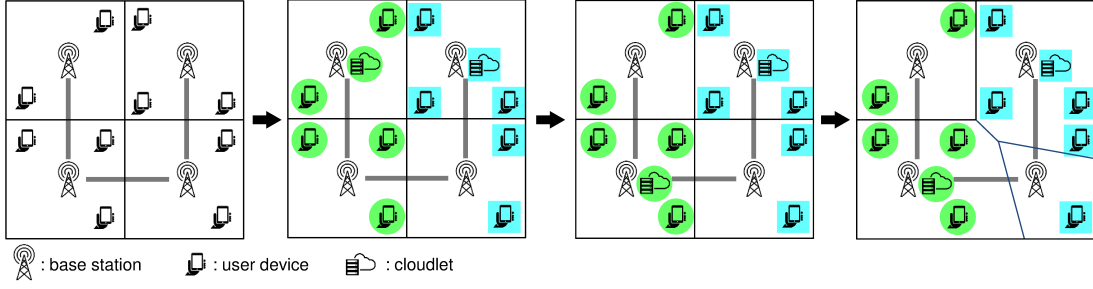


Figure 4.1: How the proposed algorithm works by assigning users to cloudlets with kMC and then using PSO to adjust the transmission power levels. © 2019 IEEE

### 4.3.3 Proposed Algorithm

The main idea behind our deployment solution is utilizing PSO to decide the physical associations and using kMC to decide the virtual associations. By choosing transmission power levels, PSO can directly control transmission delay. Additionally, by deciding which cloudlet the users send their tasks to, kMC can control processing delay. Finally, both algorithms try to lower backhaul delay. PSO can do it by coinciding physical and virtual associations in the same base station or at least associating a user physically to a base station that is close to the virtual association of that user. kMC does it by clustering users based on the resulting backhaul propagation of the virtual association. Thus, we can control all aspects of the service delay. The proposal works by first setting all base stations with the same transmission power level. This is done so we have physical associations to start working with. Then, it uses kMC to select the virtual associations and, at the end of each kMC iteration, PSO to determine the transmission power levels. This is done in every kMC iteration. By the end of the algorithm, the output is the configuration (deployment locations for cloudlets, virtual associations for users and transmission power levels for base stations) corresponding to

the iteration that received the best performance according to Equation (2.26).

An illustration of how the algorithm works can be seen in Figure 4.1. The figure first shows the location of the base stations and the users. Then, kMC is utilized by first deploying the cloudlets in random base stations. Users are assigned to the cloudlets based on the backhaul propagation needed to reach them while at the same time keeping the workload balanced between the servers. Users have the same icon (circle or square) as their corresponding cloudlets. Then, kMC moves the cloudlets to the base station closest to the users in their clusters. This is shown by moving the cloudlet with the circle icon to a different base station. With virtual associations decided, PSO is used to finally determine the transmission power levels and change some physical associations. It can be seen in the figure how this allows for keeping one user with the square icon from having to use the backhaul links to access its cloudlet (which means zero backhaul delay for this user and more bandwidth for the users that much use those links). This also means we can reduce the transmission power level of the bottom right base station, which in turn means less interference overall in the system. Both measures improve the overall transmission and backhaul delay of the users.

The full algorithm can be seen in Algorithm 4.  $\mathbb{R}^{\text{kMC}}$  is the total number of kMC iterations we want to run.  $\mathbb{K}$  is an auxiliary set we use to represent all cloudlets that still have not reached our desired amount of users.  $\mathbb{V}$  is another auxiliary set we use, this time to contain all base stations that do not have a cloudlet co-located with them yet. This can be seen in lines 15 through 17, where the cloudlets (centroids) are moved to the base station closest to the center of its cluster of associated users. Since there cannot be two cloudlets in the same base station, as per our assumptions, once a cloudlet is moved to a base station  $v_i$ , then  $v_i$  is removed from  $\mathbb{V}$  to guarantee that no other cloudlet is moved there.

---

**Algorithm 3 -PSO:** implementation for determining transmission power levels for the base stations.

---

```

1: for all particles  $\mathbb{P}$  do set  $q_{\mathbb{P}}$  with random  $\mathfrak{V}$  values
2: end for
3: for all particles  $\mathbb{P}$  do set  $\mathbb{V}_{\mathbb{P}}$  with random  $\mathfrak{V}$  values
4: end for
5: for all particles  $\mathbb{P}$  do  $\mathbb{B}_{\mathbb{P}}^l \leftarrow q_{\mathbb{P}}$ 
6: end for
7:  $\mathbb{B}^g \leftarrow \arg \min_{\text{particles } \mathbb{P}} f(\mathbb{B}_{\mathbb{P}}^l)$ 
8: for  $\mathbb{R}^{\text{PSO}}$  iterations do
9:   for all particles  $\mathbb{P}$  do
10:    if  $f(q_{\mathbb{P}}) < \mathbb{B}_{\mathbb{P}}^l$  then  $\mathbb{B}_{\mathbb{P}}^l \leftarrow q_{\mathbb{P}}$ 
11:    end if
12:    if  $f(q_{\mathbb{P}}) < \mathbb{B}^g$  then  $\mathbb{B}^g \leftarrow q_{\mathbb{P}}$ 
13:    end if
14:     $\mathbb{F}^l \leftarrow \text{random number between 0 and 1}$ 
15:     $\mathbb{F}^g \leftarrow \text{random number between 0 and 1}$ 
16:     $\mathbb{V}_{\mathbb{P}} \leftarrow \mathbb{W}^l \cdot \mathbb{V}_{\mathbb{P}} + \mathbb{F}^l \cdot \mathbb{W}^l \cdot (q_{\mathbb{P}} - \mathbb{B}_{\mathbb{P}}^l) + \mathbb{F}^g \cdot \mathbb{W}^g \cdot (q_{\mathbb{P}} - \mathbb{B}^g)$ 
17:     $q_{\mathbb{P}} \leftarrow q_{\mathbb{P}} + \mathbb{V}_{\mathbb{P}}$ 
18:  end for
19: end for
20: return  $\mathbb{B}^g$ 

```

---

## 4.4 Performance Comparison of Different Policies

In this section, we will present results to support our claim that our proposal is a valid solution for the ECC deployment problem. Our objective is to prove that our algorithm brings substantial performance improvements when compared to the random placement of servers (which is the conventional method taken in most of the literature). The results shown here are obtained through simulation. To achieve statistical reliability, the results shown are an average of 1000 different random simulations (unless stated otherwise), with each simulation having distinct locations randomly chosen for base stations and users. The backhaul links

---

**Algorithm 4 -kMC:** proposed algorithm for deploying cloudlets and configuring associations.

---

```

1: for all  $v_i \in V$  do  $\omega_{v_i} \leftarrow$  base transmission power level
2: end for
3: for  $\mathbb{R}^{\text{kMC}}$  iterations do
4:    $\mathbb{V} \leftarrow V$ 
5:   for all cloudlets  $k$  do
6:     Choose random  $v_i \in \mathbb{V}$ 
7:     Deploy  $k$  in  $v_i$ 
8:     Remove  $v_i$  from  $\mathbb{V}$ 
9:   end for
10:  repeat
11:     $\mathbb{K} \leftarrow$  all cloudlets
12:    for all  $u_i \in U$  do
13:       $z_{u_i} \leftarrow k \in \mathbb{K}$  with min. propagation to  $h_{u_i}$ 
14:      In case of ties, choose based on  $d_k^{u_i}$ 
15:      if  $z_{u_i}$  has  $\mathcal{U}/\mathcal{K}$  users then remove  $z_{u_i}$  from  $\mathbb{K}$ 
16:      end if
17:    end for
18:     $\mathbb{V} \leftarrow V$ 
19:    for all cloudlets  $k$  do
20:      Move  $k$  to  $v_i \in \mathbb{V}$  closest to its users
21:      Remove  $v_i$  from  $\mathbb{V}$ 
22:    end for
23:  until no cloudlet changes location
24:  Execute PSO for setting transmission power levels
25: end for
26: return configuration with best performance

```

---

are decided based on distance and, as mentioned before, form a passive optical network. The propagation distance is determined by the distance between the base stations, while the speed is taken as the speed of light. Additionally, the simulations utilize as parameters the values shown in Table 3 unless explicitly stated otherwise. These values follow estimates of what should be expected in 6G IoT networks [129, 130]. We assume values are the same in the uplink and the downlink. The parameters for the machine learning algorithms were obtained

Table 4.2: Parameters used in the simulations.

User average task rate	1 task/s
Average task execution time	50 ms
Number of processors per cloudlet	16
Path loss floating intercept	75.85 dB
Average path loss exponent	3.73
Total area size	10000 m <sup>2</sup>
Number of base stations	10
Number of cloudlets	3
Number of users	10000
Base station antenna gain	24.5 dBi
Rayleigh fading coefficient	-1.59175
User transmission power	27 dBm
User antenna gain	2.15 dBi
Noise density	$4 \cdot 10^{-19}$ W/Hz
Wireless channel bandwidth	1 THz
Packet size	128 KB
Data rate per backhaul link	10 Gb/s
Number of kMC iterations	6
Number of PSO iterations	6
Number of PSO particles	7
PSO inertia bias	2
PSO local best bias	18.5
PSO global best bias	2.5

from experimentation aimed at what works best for this specific problem.

Different scenarios will be evaluated to show how our proposal has the best performance. For comparison we will utilize another deployment policy conventionally seen in the literature, where cloudlets are randomly deployed, independently of the location of users, in any available base station. Virtual associations are decided based on backhaul propagation. PSO is utilized to decide transmission power levels and physical associations. Additionally, cloudlets also will serve  $\mathfrak{U}/\mathfrak{K}$  users each (i.e. balanced workload among cloudlets) in the random policy. These last two points mean that the differences in performance come strictly from

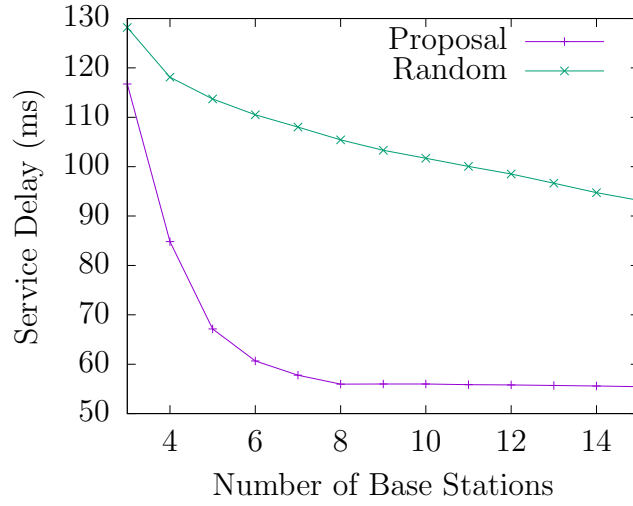


Figure 4.2: MEC service delay variation as the number of base stations is increased. © 2019 IEEE

the deployment decisions.

For the first graph, we will vary the number of base stations. The results can be seen in Figure 4.2. As expected, more base stations lead to lower service delay for all policies. This is reasonable as adding base stations means a lower communication workload per base station which consequently lowers transmission delay. Moreover, scenarios with few base stations often mean those base stations will need a high transmission power level to reach isolated users, which leads to high transmission delay due to longer propagation and poor signal for those users and higher interference across the whole system. Conversely, more base stations mean less isolated users. The random deployment policy has the worst performance, as it has no way to position cloudlets in the best base stations. Our kMC-based proposal actually chooses the best locations to deploy the cloudlets and that is where the relevant difference in performance comes from. This difference can be observed regardless of whether many or few base stations are in the system.

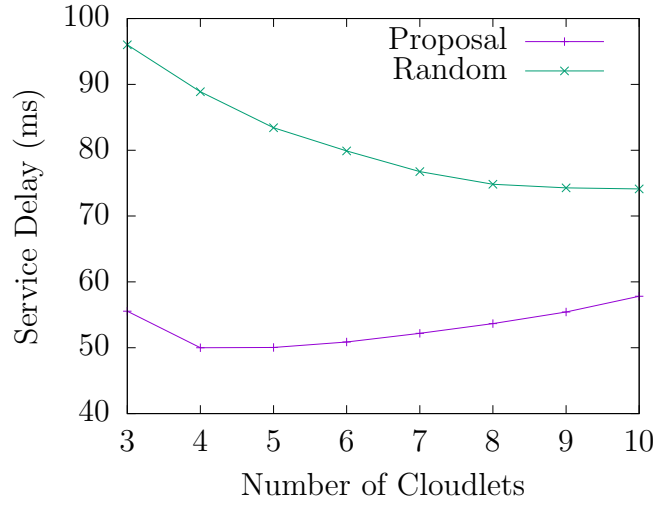


Figure 4.3: MEC service delay variation as the number of cloudlets is increased.  
© 2019 IEEE

Our proposal is also capable of efficiently using the resources in scenarios where there are few base stations such that adding new base stations brings insignificant improvements.

Next, we will analyze the policies as the number of servers increase. Results are shown in Figure 4.3. An interesting behavior can be seen where adding servers after 5 for our proposal leads to the performance becoming worse, despite the system having more resources. Adding cloudlets improves the service by lowering the processing workload per cloudlet, which leads to lower queue times for the processors as there are fewer users per cloudlet. However, if you have enough cloudlets, this wait time is so low that it becomes irrelevant. Additionally, adding cloudlets beyond that worsens the service because servers are forced to all have  $\mathcal{U}/\mathcal{K}$  users. Given this, more cloudlets mean a higher chance that a user will be forced to connect to a server that is farther away since the closest server is at the pre-determined limit already, i.e. adding more cloudlets after

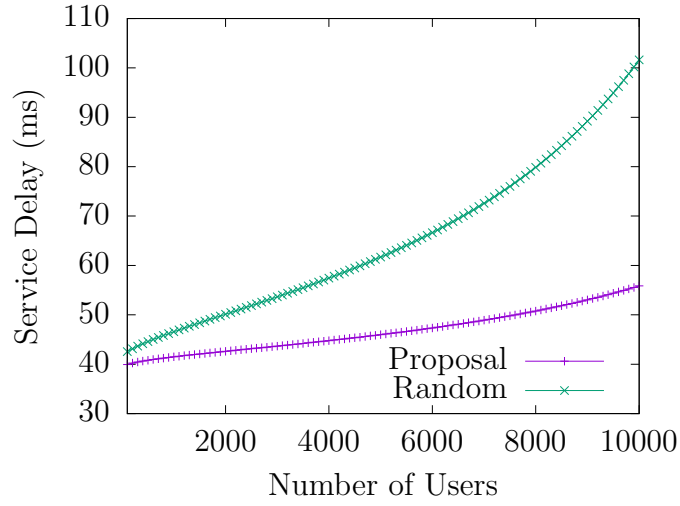


Figure 4.4: MEC service delay variation as the number of users is increased. © 2019 IEEE

queue time is already low does not improve processing time and in fact, makes transmission time worse. In this scenario, there are not enough cloudlets for the random deployment policy to experiment this behavior. The graph also shows how, again, the random policy is worse for not selecting where to deploy the cloudlets. The proposal is better regardless of how many cloudlets are in the scenario.

Finally, a comparison is made while the number of users is varied. The results are presented in Figure 4.4. As expected, more users result in worse service delay all around. This is easily explained by how users are sharing the same resource pool, both in communication and computation. Thus, more users mean fewer resources per user and consequently longer service delays. This is worst in the random policy, as there is no deployment choice to counterbalance the higher workload. kMC can position the servers according to the user locations, which results in a lower increase to the service delay, as seen in the graph.

## 4.5 Summary

MEC is a very important technology for providing resources to users. It allows for those resources to be used cheaply on-demand and shared among different clients. Moreover, differently from conventional cloud computing, the latency between users and servers is lower so even real-time applications can make use of the cloud resources. Thus, MEC is a very important technology. Because in MEC, servers are deployed in the edge instead of a single core place, the location of the deployment matters and affects the resulting service quality. Despite this, most MEC research disregards deployment and presumes a random deployment policy.

In this chapter, we proposed a deployment policy for 6G IoT environments that utilizes machine learning algorithms PSO and kMC. The proposal takes consideration of processing, transmission, and backhaul communication to effectively lower service delay. Our proposal has a low complexity and execution time and can be applied to scenarios with many variables. This is a consequence of the machine learning algorithms and it is essential for utilizing MEC with IoT and 6G that comes with massive amounts of users, servers, and requests. Moreover, the simulations showed that the proposal is significantly better than random deployment. This should prove that deployment decision is important for improving service delay in MEC and that considering transmission, processing, and backhaul communication are all relevant elements to consider when designing such a solution. The service is relevantly slower if deployment is not done properly and our proposal is capable of offering a faster service by intelligently selecting where to put each cloudlet according to user requirements.

## Chapter 5

# Cloudlet Activation Method in Dynamic MEC

### 5.1 Introduction

The main obstacle to the low Service Delay 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 [131]. 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 Quality of Service 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 consider 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 [132]. 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 [21].

Therefore, in this chapter, we will utilize the analytical model of the Service Delay in MEC from Chapter 2 (which encompasses Transmission Delay, Processing Delay, and Backhaul Delay) together with Particle Swarm Optimization (PSO, an Artificial Intelligence algorithm [109]) 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/Quality of Service restrictions. To realize and enable the configuration calculated by our PSO algorithm, we utilize Transmission Power Control [121, 122] and Virtual Machine Migration [64, 112, 113] to manage the workload and parameters of the cloudlets.

The contents of this chapter refer to the following paper, which was written based on our own research.

- T. G. Rodrigues, K. Suto, H. Nishiyama, N. Kato and K. Temma, "Cloudlets Activation Scheme for Scalable Mobile Edge Computing with Transmission Power Control and Virtual Machine Migration," in *IEEE Transactions on Computers*, vol. 67, no. 9, pp. 1287-1300, September 2018. © 2011 IEEE

## 5.2 History of Improving MEC 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 Virtual Machine 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 Virtual Machine servers in the same cloudlet have to share the same resource pool, Processing Delay tends to grow along with the number of Virtual Machines contending for such resources (in an effect called multi-tenancy [133]). Thus, if efforts are made to avoid cloudlets hosting more Virtual Machine servers than necessary, it should be effective in managing Processing Delay and improving computation scalability. As a result, balancing the amount of Virtual Machine servers hosted by each cloudlet to intelligently use computation resources is often seen in the literature. S. Roy *et al.* [110] and J. Oueis *et al.* [111] balanced the workload by always setting up the Virtual Machine servers of new incoming users in the cloudlet with the least amount of Virtual Machines; this guarantees that the difference between the amount of hosted Virtual Machine servers among the cloudlets is always minimal. L. Gkatzikis and I. Koutsopoulos [112], and M. Mishra *et al.* [113] achieve the same goal by using Virtual Machine Migration to move Virtual Machines between cloudlets. If the migra-

tion is done from overworked cloudlets to ones hosting few Virtual Machines, then this would improve the efficiency of cloudlet usage by avoiding situations where cloudlets are idle with wasted capacity. M. Jia *et al.* [134] offer a different insight on Virtual Machine 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). S. Farrugia [116], and X. Zhu *et al.* [117] 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. Y. Li and W. Wang [118], and K. Suto *et al.* [119] 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. L. Yang *et al.* [120]

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, G. von Zengen *et al.* [121], and T. Aota and K. Higuchi [122] 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. M. Peng *et al.* [123] 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 [76]. 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. Y. Mao, J. Zhang and K. B. Letaief [135] 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. S. Sardellitti *et al.* [136] 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. X. Chen *et al.* [20], Y. Yu *et al.* [135], and Wang *et al.* [79] also simultaneously consider Transmission Delay and Processing Delay under the same scenario restrictions. Meanwhile, Y. Wang *et al.* [137], T. Q. Dinh *et al.* [62], and K. Sato and T. Fujii [138] 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 [54]. 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 Virtual Machine 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.

Finally, our proposal is focused on MEC and computation offloading to cloudlets,

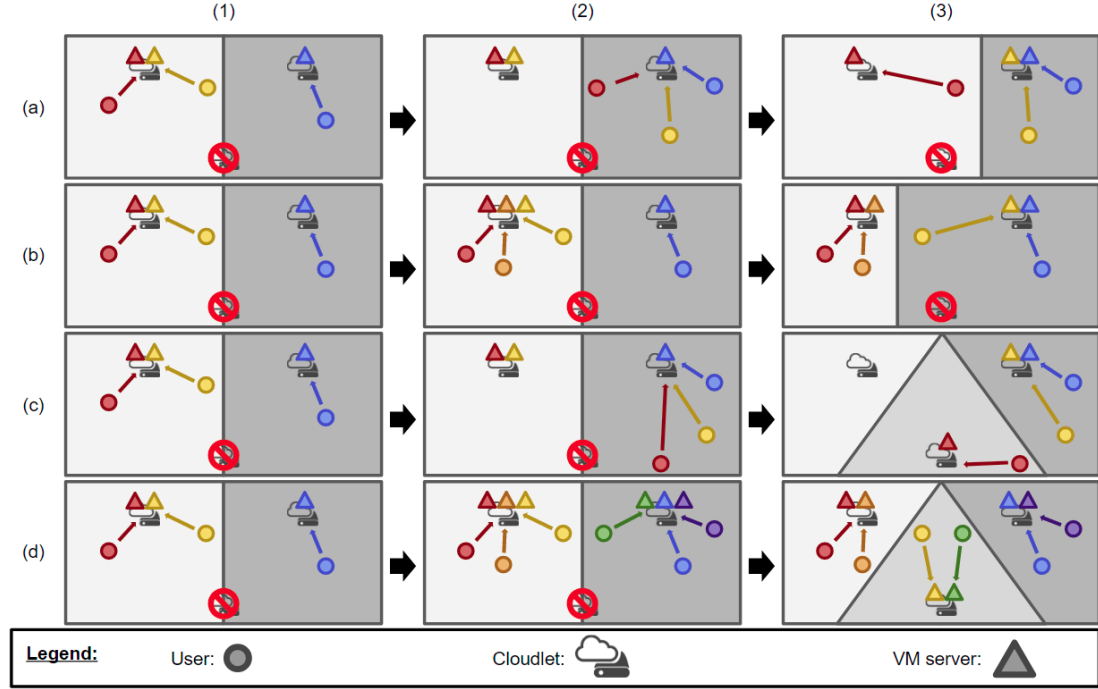


Figure 5.1: Illustrative scenarios of how user mobility ((a) and (c)) and new users ((b) and (d)) can increase Service Delay and how cloudlet activation ((a) and (b)) and reconfiguration ((c) and (d)) can counteract this. © 2018 IEEE

but there are other variations of distributed computing where instead of cloudlets the tasks are offloaded to other user equipments (as in Fog Computing [24] 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.

### 5.3 Artificial Intelligence for Activating Cloudlets and Configuring MEC

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 Virtual Machine Migration [64, 112, 113] and Transmission Power Control [121] to reconfigure the system to maximize scalability. Because Virtual Machine Migration controls virtual associations (by moving Virtual Machine 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.

---

**Algorithm 5 - 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: end while
5: return  $\hat{t}_k$ 

```

---

Figure 5.1 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 Virtual Machines. 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 5; by delaying as much as possible the next violation of the 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 (2.26), 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

---

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

---

power levels. Input  $g$  is a tuple with  $|C|$  values. However, the problem of finding the input that maximizes the output of Algorithm 5 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.

PSO [109] 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

---

**Algorithm 7 - 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:     end while
10:  else
11:    randomly activate one inactive cloudlet
12:  end if
13: until no more cloudlets left to activate

```

---

locally (by that particle), and the best solution 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 ( $\vartheta$ ), local acceleration ( $\varrho_h$ ), and global acceleration ( $\varrho_g$ ) 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 [125]. 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  $\varrho_h$  and  $\varrho_g$ ). 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 5. The PSO algorithm we utilize is shown in pseudo-code in Algorithm (6), 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

(6) 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 7, 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 (6) returns as second output that it cannot delay Configuration Phase beyond the current time.

Because of our choice of Algorithm 5 as a fitness function, the configuration found by the particles maximizes scalability and avoids cloudlet activation. Furthermore, because line 2 in Algorithm 5 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 the previous section.

## 5.4 Proposal 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.

Table 5.1: 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 <sup>2</sup>		
User arrival rate	1 user/(km <sup>2</sup> *s)		
Total area size	0.01 km <sup>2</sup>		
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 <sup>-19</sup> W/Hz		
Short distance path loss coefficient	2		
Long distance path loss coefficient	4		
Path loss distance breakpoint	100m		
Wireless propagation speed	3*10 <sup>8</sup> 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		

#### 5.4.1 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 ana-

lytical 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 [139]. 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 7. If the Service Delay levels are above what is allowed even after 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.

### 5.4.2 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

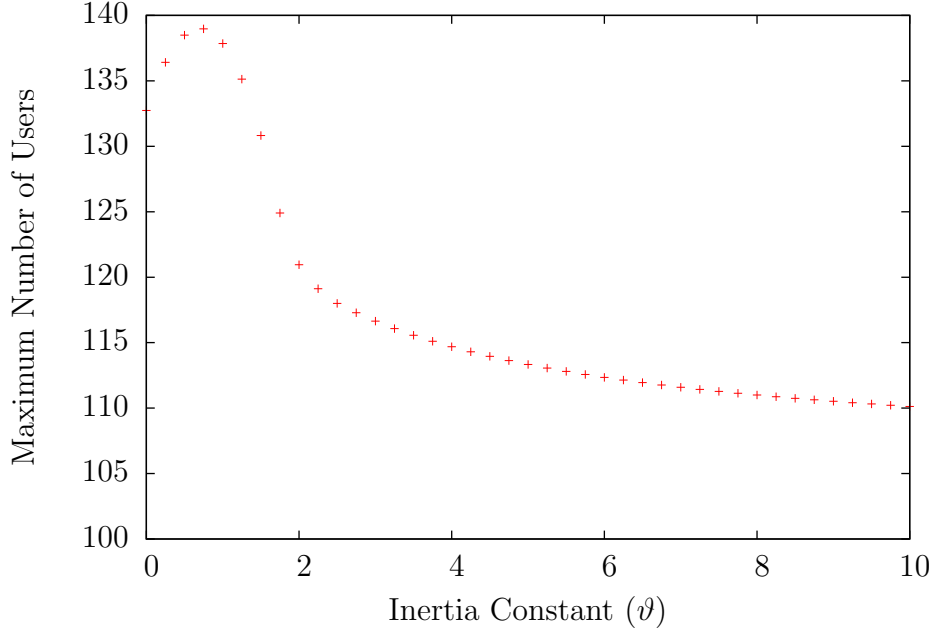


Figure 5.2: Maximum number of users served by our proposal with five cloudlets under different values for PSO local inertia ( $\vartheta$ ). © 2018 IEEE

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

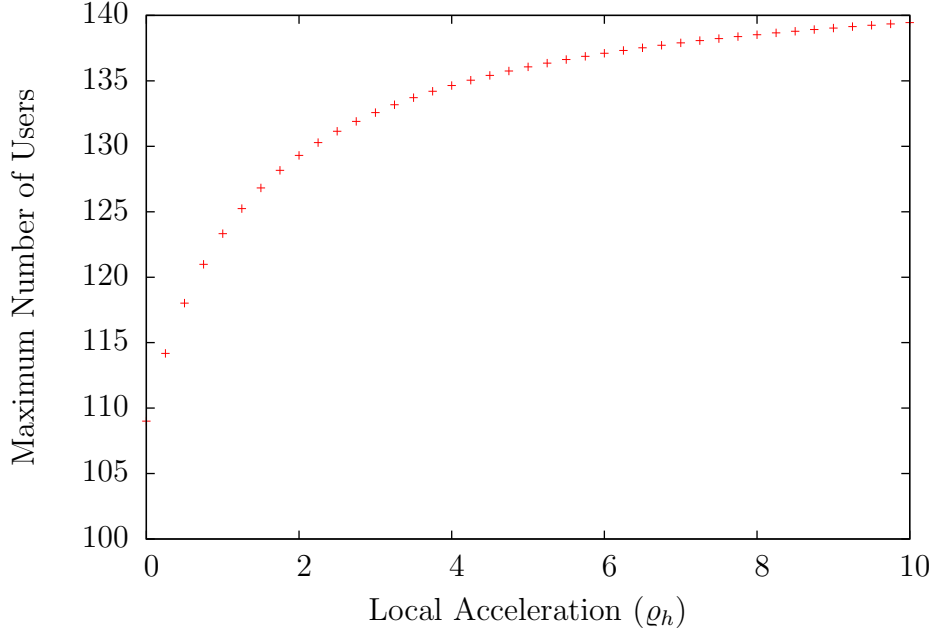


Figure 5.3: Maximum number of users served by our proposal with five cloudlets under different values for PSO local acceleration ( $\varrho_h$ ). © 2018 IEEE

random search in the domain of possible solutions. However, as Figure 5.2 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 [140]. 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

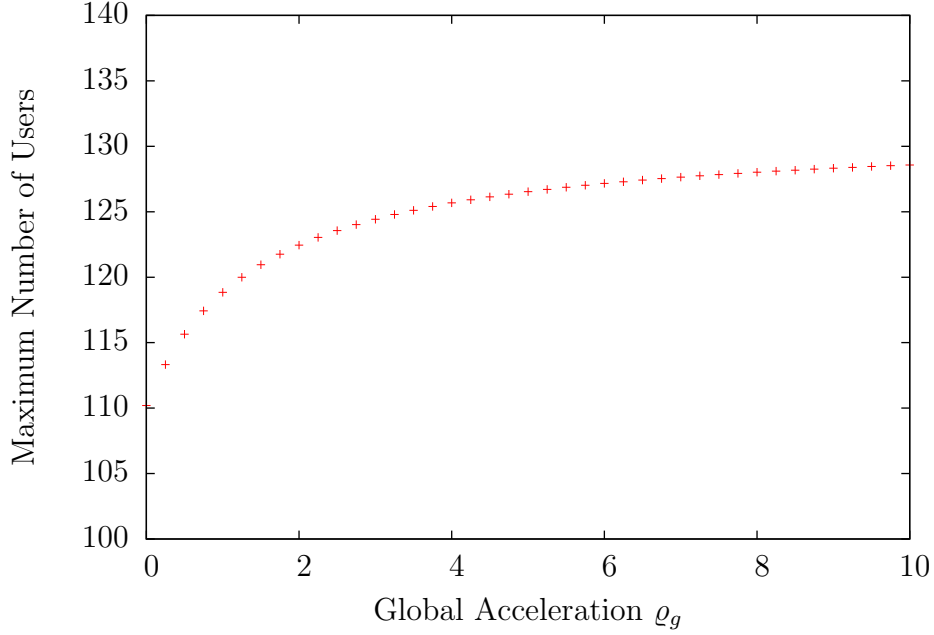


Figure 5.4: Maximum number of users served by our proposal with five cloudlets under different values for PSO global acceleration ( $\varrho_g$ ). © 2018 IEEE

solution found by that particular particle and all particles respectively. The results are shown in Figure 5.3 and Figure 5.4. 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 [141]. 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.

### 5.4.3 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 [118], 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 [112], where Virtual Machine 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 [134], 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

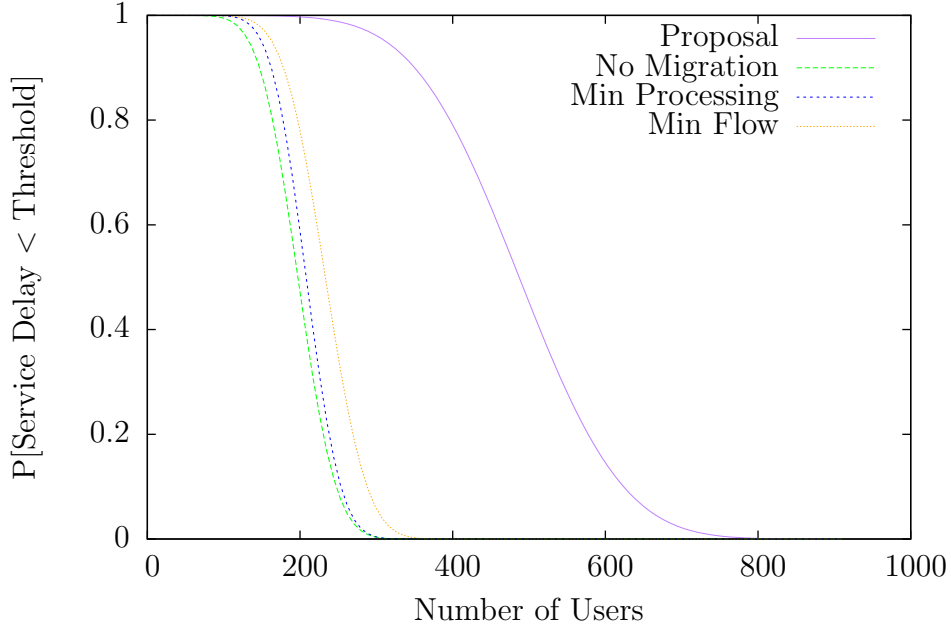


Figure 5.5: Probability that latency is below the threshold based on the number of users using Application A (no relevant difference between communication or computation burdens). © 2018 IEEE

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).

Figure 5.5 contains the performance of all methods in terms of what the probability is that the latency is under the 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

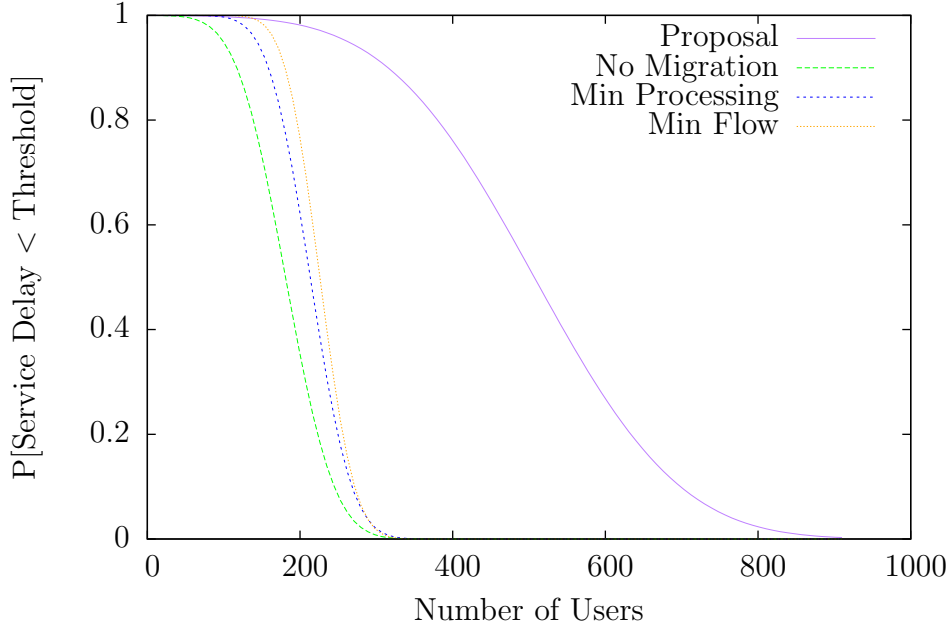


Figure 5.6: Probability that latency is below the threshold given the number of users using Application B (heavier burden on processing). © 2018 IEEE

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 Figure 5.6, because of the higher burden on computation from Application B, the Minimum Processing 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

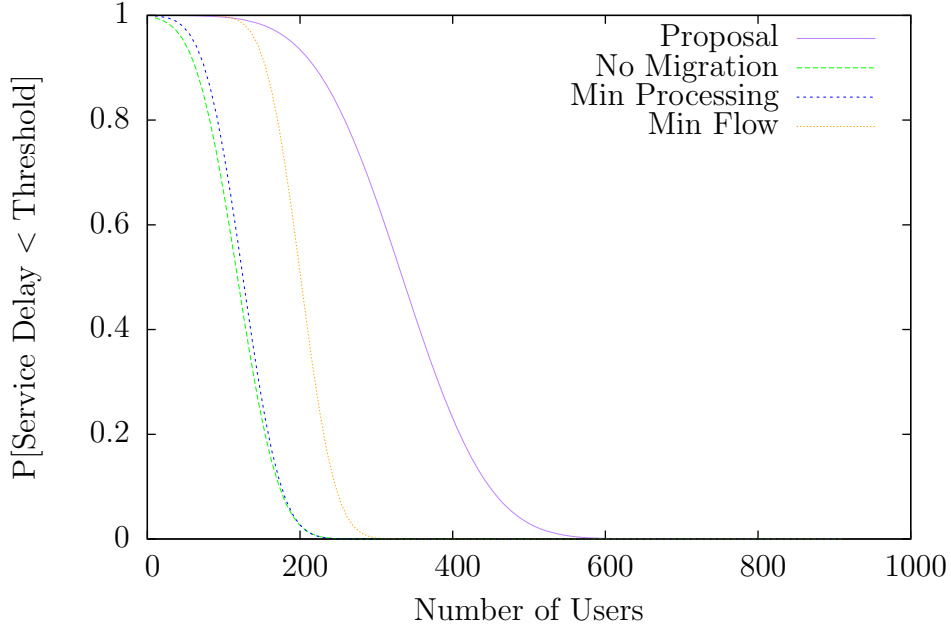


Figure 5.7: Probability that latency is below the threshold given the number of users using Application C (heavier burden on transmission). © 2018 IEEE

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 Figure 5.7. This can

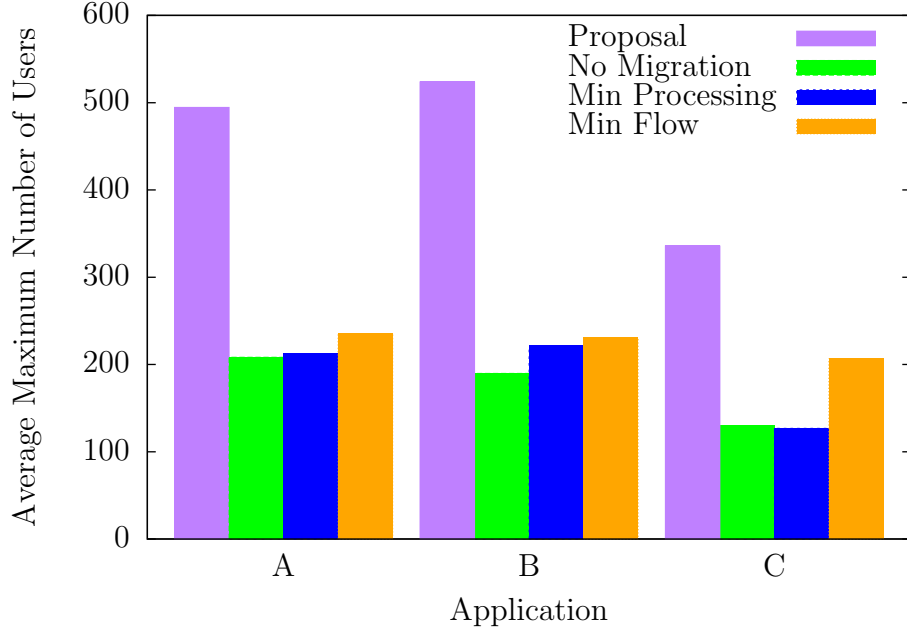


Figure 5.8: Maximum number of users for all methods under all applications. © 2018 IEEE

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 Figure 5.8, 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.

## 5.5 Summary

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 chapter, 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.

# Chapter 6

## Conclusion

Mobile Edge Computing (MEC) is an essential technology for future networks. Its main use and attractiveness is how it provides powerful and important cloud resources to users from the edge of the network. Those resources allow mobile devices to execute demanding applications in spite of their local limitations. Additionally, because this is a cloud model, the resources are offered in demand, when they are needed and as much as they are needed. Since the use of the cloud server is shared between users, it is a cheap service model that is accessible to users in general. Also of note is how MEC has the servers in the edge of the network. This translates to low latencies that cannot be offered by conventional cloud computing systems where the servers are far away from the users. Consequently, MEC can be utilized even with real-time applications and services that demand low delay. To make this possible, in MEC there are many servers so that no user is too far from a server and low latency is always possible. To counterbalance this economic cost, the servers in MEC are less powerful than conventional cloud servers, earning their moniker of cloudlets. Because of all these advantages, MEC is expected to be one of the supporting technologies in Internet of Things (IoT) and 5G/6G networks. This comes with major difficulties however. First

of all, MEC has the aforementioned high number of servers and also a very high number of user devices and base stations because of its use in IoT and 5G/6G. The problem is that all this elements make the MEC models too complex and with too many variables. Thus, conventional, heuristic algorithms/protocols are not applicable to solve and configure MEC as they would too long to execute. Furthermore, MEC is utilized in the edge of the network with wireless devices and environments. Consequently, it is subject to a highly dynamic scenario and thus its protocols must be executed quickly and multiple times during the lifetime of the service. Because of these complications, we envision that Machine Learning solutions are the best method for designing protocols for MEC. Under this paradigm, the algorithms look for the best solutions by analyzing the patterns of the problem and exploring different solutions to learn what works for each problem. Machine Learning allows us to find near optimal configurations at very low complexity and execution time. This is possible even in situations with too many variables and data. With this in mind, in this thesis we provide different protocols to configure MEC. Each protocol utilizes Machine Learning elements to solve important MEC problems.

In Chapter 2, we presented an in-depth analysis of MEC based on existing literature. From this analysis, we surmised a basic service model where users generate tasks that are taken to cloudlets for execution through a system of wireless connections and wired backhaul links. Then, the cloudlets send the back the results of such tasks to the users. Also of note is how each user has a corresponding Virtual Machine server located in one of the cloudlets of the system. This Virtual Machine server is useful for compartmentalizing resources and more easily keeping data related to the corresponding user. From the literature we also note how two metrics are very important in MEC: service delay and user capacity. Service delay is the time between the generation of a task and the

arrival of its results. Low delay is essential because it satisfies the real-time applications, allows users to leave the system quicker, and releases resources to be used with the other users. Thus, it is important to configure MEC to finish the tasks as fast as possible. User capacity is how many users can be connected to each cloudlet while still respecting the requirements of each user and the resource limitations of the servers. High user capacity is important because more users means a higher profit to service providers which consequently results in lower costs for users. Obviously, this should be done without compromising quality of service. Based on these findings, we provide a mathematical model for estimating the service delay and user capacity of a MEC system given its specifications. The model is prepared to take into account user mobility and also processing delay, transmission delay and backhaul. Thus, we can it realistically represents MEC scenarios and it is very useful for designing solutions.

In Chapter 3, we provide a method to allocate MEC resources to users in order to minimize service delay. Resource allocation is done by deciding which base station and which cloudlet each user will connect to. This allows us to balance the workload between base stations and cloudlets and guarantee that their resources are efficiently used and never idle. To decide these connections, we individually decide the transmission power level of each MEC, thus controlling which base station provides the highest signal to each user. Additionally, we migrate Virtual Machine servers between cloudlets to decide which users are connected to each one. To decide which transmission power level to set and how to migrate the servers, we utilize a Machine Learning algorithm called Particle Swarm Optimization. The algorithm efficiently explores configurations and exploits the solutions with better performance. Thus, we propose a protocol that uses Particle Swarm Optimization to find a configuration with minimum service delay. Our proposal using this algorithm is shown to find near-optimal solutions at a small

fraction of the time taken by conventional solutions to execute. Moreover, the proposal is compared against other conventional methods and it provides lower latency. This result holds even if different application profiles. With this, we prove how Machine Learning is efficient for MEC and also that considering transmission delay and backhaul delay is essential for lowering service delay (compared to conventional methods that only optimize processing delay).

In Chapter 4, we propose a solution for deciding where to locate the cloudlets in a MEC system. Contrary to conventional cloud systems, where the servers are also put together in a single location, the cloudlets in MEC are spread around the edge of the network. The location of the cloudlets can significantly impact the overall service delay of the system. In our proposal, we presume that the cloudlets must be co-located with base stations, to minimize the backhaul delay between base stations and cloudlets. Thus, we designed a system to decide which base stations should receive cloudlets. We realize this by utilizing k-Means Clustering, which is a Machine Learning algorithm. k-Means Clustering is able to cluster elements efficiently which trying to optimize an objective function. This is done by exploring different random locations of cluster centers, clustering users to closest centers and adjusting the position of these centers based on the features of the elements associated to it. This cycle is repeated as the algorithm looks for the best solution. Our proposal thus uses k-Means Clustering to cluster users and then deploys one cloudlet in the center of each cluster of user. Then, it utilizes the protocol from Chapter 3 to allocate resources. This results in a significantly lower delay when compared to random cloudlet deployment, which is the standard method found in the literature. This result stands even with different amounts of base stations, users and cloudlets.

In Chapter 5, we presume a system where cloudlets are already intelligently deployed among base stations. Then, we proposal a protocol that decided which

cloudlets should be turned on and which should be left turned off. This is important because turning on all cloudlets may not be necessary depending on how many users are connected. Having unnecessary cloudlets turned on will increase the operational expenses of the system, thus reducing profits and increasing the price of the service. Obviously, our protocol is designed to decide which cloudlet to be turned on without compromising in terms of quality of service. Thus, the protocol must keep enough cloudlets working such that the service delay requirements of the users are respected. We once more use Particle Swarm Optimization as a Machine Learning solution, thus guaranteeing that we find near-optimal configurations with a low complexity. Our proposal is capable of deciding which cloudlets to turn on despite users joining the system constantly, moving inside the area and connecting to different base stations/cloudlets. Our proposal, when compared to conventional solutions from the literature, is capable of service significantly more users per cloudlet while respecting the service requirements. Depending on the application profile, our proposal can serve more than twice as many users.

Finally, in Chapter 6, we offer concluding remarks to this thesis. The most important results of this research are a mathematical model to properly represent MEC in a realistic way, including the whole service stack and service delay and the protocols to configure MEC. These protocols are all built based on Machine Learning and consequently are capable of handling very high amounts of users, base stations and cloudlets while still executing quickly. From the performance of our protocols, it is clear that our solutions are relevantly better than what it already existis in the literature. Additionally, Machine Learning is possibly the only method to handle the high amount of users/base stations/cloudlets that should exist in the future networks. We expect that this type of solutions will be the standard for network design in the future.



## Acknowledgments

I would like to start by thanking the immense help and support offered by my family, both in Japan and Brazil. My father Claudio Rodrigues, my mother Beatriz Gama Rodrigues and my brothers André Luís Gama Rodrigues and João Pedro Rodrigues, as well as my mother-in-law Kuniko Koketsu, my brother-in-law Masuhiro Koketsu and my sister-in-law Masayo Yamazaki. Ever single one of these have offered me a home and a place to relax, both physically and mentally, and thus have been an emotional fortress during these years.

I am deeply grateful to my academic advisor, Prof. Nei Kato, for giving me this opportunity. I would not even be in this country if it was not for him, and his guidance during this research is what made it possible for me to produce. Learning under him was an incredibly period in my life.

I would like to express my gratitude to Prof. Takuo Suganuma, Prof. Hiroki Nishiyama and Prof. Yuichi Kawamoto for composing the supervisory committee of this thesis and offering their time, attention and guidance to improve this research. Through their advices, I have greatly improved the quality of my work.

I also thank Prof. Katsuya Suto and Shikhar Verma for their discussions regarding my research. Their support helped keep me in the right track and their feedback enriched my research immensely. I am also thankful to Motoko Shiraishi, Kaoru Chiba, Prof. Zubair Md. Fadlullah and Takako Kase for offering support in many opportunities during my Doctor's Course.

I gratefully acknowledge my friends for their support and companionship during these difficult years. I want to mention João Paulo Apolinário Passos, Rafael Viana Ribeiro, Athos Silva, Cassiano Cerqueira, Emanuel Abreu, Iure Rebouças, Lucas Rodrigues, Paula Costa do Valle, and Warley Franciso Ribeiro. They were always there to listen to me, no matter how stressed or tired I was. I would not be where I am without them and I will always be thankful for their friendship.

Acknowledgments are also given to the Ministry of Education, Culture, Sports, Science and Technology of Japan and the Japan Society for the Promotion of

## Acknowledgments

---

Science (JSPS). Their financial support is what allowed me to do the research shown here and greatly diminished my worries and allowed me to focus on my work as a researched.

Finally, I want to thank my wife and partner, the love of my life, Ami Koketsu, for being there for me all this time. Not a single step of this project would have been possible without her. Her love, care, patience and attention are what drive me forward and keep me up in the best and worst days. I will strive for the rest of my life to repay all that she has given me and it will not be enough. I love you, Ami.

And, as always, I thank God All Mighty for all the opportunities and results in my whole life. Thank you for taking me where I am today and allowing me to do all these things.



# Bibliography

# Bibliography

- [1] R. Buyya, C. S. Yeo, and S. Venugopal, “Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities,” in *Proceedings of the 2008 10th IEEE International Conference on High Performance Computing and Communications*, September 2008, pp. 5-13.
- [2] Q. Ding, B. Tang, P. Manden, and J. Ren, “A Learning-Based Cost Management System for Cloud Computing,” in *Proceedings of the 2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*, January 2018, pp. 362-367.
- [3] L. Columbus. (2017, October) Cloud Computing Market Projected To Reach \$411B By 2020. Accessed: 2018-03-01. [Online]. Available: <https://www.forbes.com/sites/louiscolombus/2017/10/18/cloudcomputing-market-projected-to-reach-411b-by-2020/#370bda7278f2>
- [4] Google Cloud Platform: Google Cloud Computing, Hosting Services and API. Accessed: 2018-03-01. [Online]. Available: <https://cloud.google.com/>
- [5] Amazon Web Services (AWS) - Cloud Computing Services. Accessed: 2018-03-01. [Online]. Available: <https://aws.amazon.com/>
- [6] Microsoft Azure Cloud Computing Platform and Services. Accessed: 2018-03-01. [Online]. Available: <https://azure.microsoft.com/>
- [7] K. Kumar and Y. H. Lu, “Cloud Computing for Mobile Users: Can Offloading Computation Save Energy?,” *Computer*, vol. 43, no. 4, pp. 51-56, April 2010.

- [8] S. Al-Janabi, I. Al-Shourbaji, M. Shojafar, and M. Abdelhag, “Mobile Cloud Computing: Challenges and Future Research Directions,” in *Proceedings of the 2017 10th International Conference on Developments in eSystems Engineering (DeSE)*, June 2017, pp. 62-67.
- [9] M. Satyanarayanan, “Fundamental Challenges in Mobile Computing,” in *Proceedings of the 1996 15th Annual ACM Symposium on Principles of Distributed Computing*, May 1996, pp. 1-7.
- [10] M. Lopez-Nores, Y. Blanco-Fernandez, J. J. Pazos-Arias, A. Gil-Solla, and M. Ramos-Cabrer, “Augmented Reality, Smart Codes and Cloud Computing for Personalized Interactive Advertising on Billboards,” in *Proceedings of the 2015 10th International Workshop on Semantic and Social Media Adaptation and Personalization (SMAP)*, November 2015, pp. 1-6.
- [11] A. A. Ateya, A. Vybornova, R. Kirichek, and A. Koucheryavy, “Multilevel Cloud Based Tactile Internet System,” in *Proceedings of the 2017 19th International Conference on Advanced Communication Technology (ICACT)*, February 2017, pp. 105-110.
- [12] G. PremSankar, M. D. Francesco, and T. Taleb, “Edge Computing for the Internet of Things: A Case Study,” *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 1275-1284, April 2018.
- [13] F. Wang, J. Xu, X. Wang, and S. Cui, “Joint Offloading and Computing Optimization in Wireless Powered Mobile-Edge Computing Systems,” in *Proceedings of the 2017 IEEE International Conference on Communications (ICC)*, May 2017, pp. 1-6.
- [14] R. Saadoon, R. Sakat, and M. Abbod, “Small Cell Deployment for Data Only Transmission Assisted by Mobile Edge Computing Functionality,” in *Proceedings of the 2017 6th International Conference on Future Generation Communication Technologies (FGCT)*, August 2017, pp. 1-6.
- [15] A. A. Laghari, H. He, M. Shafiq, and A. Khan, “Assessing Effect of Cloud Distance on End User ’ s Quality of Experience (QoE),” in *Proceedings of*

- the 2016 2nd IEEE International Conference on Computer and Communications (ICCC)*, October 2016, pp. 500-505.
- [16] K. Fogarty. (2013, March) In the Cloud, Distance Matters for Compute Efficiency. Accessed: 2018-03-02. [Online]. Available: <https://www.networkcomputing.com/data-centers/cloud-distancematters-compute-efficiency/1889266701>
- [17] K. Bierzynski, A. Escobar, and M. Eberl, “Cloud, Fog and Edge: Cooperation for the Future?” in *Proceedings of the 2017 2nd International Conference on Fog and Mobile Edge Computing (FMEC)*, May 2017, pp. 62-67.
- [18] H. Huang, Y. Cai, and H. Yu, “Distributed-Neuron-Network Based Machine Learning on Smart-Gateway Network Towards Real-Time Indoor Data Analytics,” in *Proceedings of the 2016 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2016, pp. 720-725.
- [19] L. Wang, O. Brun, and E. Gelenbe, “Adaptive Workload Distribution for Local and Remote Clouds,” in *Proceedings of the 2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, October 2016, pp. 3984-3988.
- [20] 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.
- [21] 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.
- [22] M. Aazam and E. N. Huh, “Fog Computing and Smart Gateway Based Communication for Cloud of Things,” in *Proceedings of the 2014 International Conference on Future Internet of Things and Cloud*, August 2014, pp. 464-470.

- [23] S. Lavanya, N. M. S. Kumar, S. Thilagam, and S. Sinduja, “Fog Computing Based Radio Access Network in 5G Wireless Communications,” in *Proceedings of the 2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, March 2017, pp. 559-563.
- [24] “IEEE Standard for Adoption of OpenFog Reference Architecture for Fog Computing,” *IEEE Standard 1934-2018*, pp. 1-176, August 2018.
- [25] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, “On Multi-Access Edge Computing: A Survey of the Emerging 5G Network Edge Cloud Architecture and Orchestration,” *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1657-1681, May 2017.
- [26] ETSI. (2019) Multi-access Edge Computing (MEC). Accessed: 2019-05-15. [Online]. Available: <https://www.etsi.org/technologies/multi-access-edge-computing>
- [27] 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.
- [28] Z. Zhang and W. Hao, “Development of a New Cloudlet Content Caching Algorithm Based on Web Mining,” in *Proceedings of the 2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*, January 2018, pp. 329-335.
- [29] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, “Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications,” *IEEE Communications Surveys Tutorials*, vol. 17, no. 4, pp. 2347-2376, June 2015.
- [30] J. Pan and J. McElhannon, “Future Edge Cloud and Edge Computing for Internet of Things Applications,” *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 439-449, February 2018.

- [31] V. Balasubramanian, N. Kouvelas, K. Chandra, R. V. Prasad, A. G. Voyiatzis, and W. Liu, “A Unified Architecture for Integrating Energy Harvesting IoT Devices with the Mobile Edge Cloud,” in *Proceedings of the 2018 IEEE 4th World Forum on Internet of Things (WF-IoT)*, February 2018, pp. 13-18.
- [32] 5G: The Internet for Everyone and Everything. Accessed: 2018-03-02. [Online]. Available: [http://www.ni.com/pdf/company/en/Trend\\_Watch\\_5G.pdf](http://www.ni.com/pdf/company/en/Trend_Watch_5G.pdf)
- [33] Internet of Things Forecast. Accessed: 2018-03-02. [Online]. Available: <https://www.ericsson.com/en/mobility-report/internet-of-thingsforecast>
- [34] E. Renart, D. Balouek-Thomert, X. Hu, J. Gong, and M. Parashar, “Online Decision-Making Using Edge Resources for Content-Driven Stream Processing,” in *Proceedings of the 2017 IEEE 13th International Conference on e-Science (e-Science)*, October 2017, pp. 384-392.
- [35] K. Intharawijitr, K. Iida, H. Koga, and K. Yamaoka, “Practical Enhancement and Evaluation of a Low-Latency Network Model Using Mobile Edge Computing,” in *Proceedings of the 2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, July 2017, pp. 567-574.
- [36] B. Ahlgren, M. Hidell, and E. C. Ngai, “Internet of Things for Smart Cities: Interoperability and Open Data,” *IEEE Internet Computing*, vol. 20, no. 6, pp. 52-56, November 2016.
- [37] U. S. Shanthamallu, A. Spanias, C. Tepedelenlioglu, and M. Stanley, “A Brief Survey of Machine Learning Methods and Their Sensor and IoT Applications,” in *Proceedings of the 2017 8th International Conference on Information, Intelligence, Systems Applications (IISA)*, August 2017, pp. 1-8.
- [38] S. Das and M. J. Nene, “A Survey on Types of Machine Learning Techniques in Intrusion Prevention Systems,” in *Proceedings of the 2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, March 2017, pp. 2296-2299.

- [39] A. L. Samuel, “Some Studies in Machine Learning Using the Game of Checkers,” *IBM Journal of Research and Development*, vol. 3, no. 3, pp. 210-229, July 1959.
- [40] J. R. Koza, F. H. Bennet III, D. Andre, and M. A. Keane, “Automated Design of Both the Topology and Sizing of Analog Electrical Circuits Using Genetic Programming,” in *Artificial Intelligence in Design '96*, J. S. Gero and F. Sudweeks, Eds. Springer, 1996.
- [41] C. Bishop, *Pattern Recognition and Machine Learning*. SpringerVerlag New York, 2006.
- [42] N. M. Nasrabadi, “Pattern Recognition and Machine Learning,” *Journal of Electronic Imaging*, vol. 16, no. 4, pp. 1-2, October 2007.
- [43] D. E. Goldberg and J. H. Holland, “Genetic Algorithms and Machine Learning,” *Machine Learning*, vol. 3, no. 2, pp. 95-99, October 1988.
- [44] O. Simeone. (2018, August) A Very Brief Introduction to Machine Learning With Applications to Communication Systems. Accessed: 2019-05-09. [Online]. Available: <https://arxiv.org/abs/1808.02342>
- [45] Z. M. Fadlullah, F. Tang, B. Mao, N. Kato, O. Akashi, T. Inoue, and K. Mizutani, “State-of-the-Art Deep Learning: Evolving Machine Intelligence Toward Tomorrow’s Intelligent Network Traffic Control Systems,” *IEEE Communications Surveys Tutorials*, vol. 19, no. 4, pp. 2432-2455, November 2017.
- [46] S. N. Shirazi, A. Gouglidis, A. Farshad, and D. Hutchison, “The Extended Cloud: Review and Analysis of Mobile Edge Computing and Fog From a Security and Resilience Perspective,” *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2586-2595, November 2017.
- [47] G. Yang, Q. Sun, A. Zhou, S. Wang, and J. Li, “Access Point Ranking for Cloudlet Placement in Edge Computing Environment,” in *Proceedings of the 2016 IEEE/ACM Symposium on Edge Computing (SEC)*, October 2016, pp. 85-86.

- [48] L. Zhao, W. Sun, Y. Shi, and J. Liu, “Optimal Placement of Cloudlets for Access Delay Minimization in SDN-Based Internet of Things Networks,” *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 1334-1344, April 2018.
- [49] C. Meurisch, J. Gedeon, A. Gogel, T. A. B. Nguyen, F. Kaup, F. Kohnhaeuser, L. Baumgaertner, M. Schmittner, and M. Muehlhaeuser, “Temporal Coverage Analysis of Router-Based Cloudlets Using Human Mobility Patterns,” in *Proceedings of the 2017 IEEE Global Communications Conference (GLOBECOM 2017)*, December 2017, pp. 1-6.
- [50] S. Kachele, C. Spann, F. J. Hauck, and J. Domaschka, “Beyond IaaS and PaaS: An Extended Cloud Taxonomy for Computation, Storage and Networking,” in *Proceedings of the 2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing*, December 2013, pp. 75-82.
- [51] G. Liu, “Research on Independent SaaS Platform,” in *Proceedings of the 2010 2nd IEEE International Conference on Information Management and Engineering*, April 2010, pp. 110-113.
- [52] R. Dua, A. R. Raja, and D. Kakadia, “Virtualization vs Containerization to Support PaaS,” in *Proceedings of the 2014 IEEE International Conference on Cloud Engineering*, March 2014, pp. 610-614.
- [53] M. Kozlovsky, M. Torocsik, T. Schubert, and V. Poserne, “IaaS Type Cloud Infrastructure Assessment and Monitoring,” in *Proceedings of the 2013 36th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, May 2013, pp. 249-252.
- [54] 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.
- [55] T. G. Rodrigues, K. Suto, H. Nishiyama, N. Kato, and K. Temma, “Cloudlets Activation Scheme for Scalable Mobile Edge Computing with

- Transmission Power Control and Virtual Machine Migration,” *IEEE Transactions on Computers*, vol. 67, no. 9, pp. 1287-1300, September 2018.
- [56] H. Li, K. Ota, and M. Dong, “Learning IoT in Edge: Deep Learning for the Internet of Things with Edge Computing,” *IEEE Network*, vol. 32, no. 1, pp. 96-101, January 2018.
- [57] Q. Fan and N. Ansari, “Workload Allocation in Hierarchical Cloudlet Networks,” *IEEE Communications Letters*, vol. 22, no. 4, pp. 820-823, April 2018.
- [58] L. Zhao and J. Liu, “Optimal Placement of Virtual Machines for Supporting Multiple Applications in Mobile Edge Networks,” *IEEE Transactions on Vehicular Technology*, vol. 67, no. 7, pp. 6533-6545, July 2018.
- [59] L. Chen, S. Patel, H. Shen, and Z. Zhou, “Profiling and Understanding Virtualization Overhead in Cloud,” in *Proceedings of the 2015 44th International Conference on Parallel Processing*, September 2015, pp. 31-40.
- [60] N. Tziritas, M. Koziri, A. Bachtsevani, T. Loukopoulos, G. Stamoulis, S. U. Khan, and C. Xu, “Data Replication and Virtual Machine Migrations to Mitigate Network Overhead in Edge Computing Systems,” *IEEE Transactions on Sustainable Computing*, vol. 2, no. 4, pp. 320-332, October 2017.
- [61] I. Farris, T. Taleb, M. Bagaa, and H. Flick, “Optimizing Service Replication for Mobile Delay-Sensitive Applications in 5G Edge Network,” in *Proceedings of the 2017 IEEE International Conference on Communications (ICC)*, May 2017, pp. 1-6.
- [62] 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.
- [63] C. Liu, Y. Cao, Y. Luo, G. Chen, V. Vokkarane, M. Yunsheng, S. Chen, and P. Hou, “A New Deep Learning-Based Food Recognition System for Dietary Assessment on An Edge Computing Service Infrastructure,” *IEEE*

- Transactions on Services Computing*, vol. 11, no. 2, pp. 249-261, March 2018.
- [64] S. Kim, X. de Foy, and A. Reznik, "Practical Service Allocation in Mobile Edge Computing Systems," in *Proceedings of the 2017 27th International Telecommunication Networks and Applications Conference (ITNAC)*, November 2017, pp. 1-6.
- [65] B. P. Rimal, D. P. Van, and M. Maier, "Mobile-Edge Computing Versus Centralized Cloud Computing Over a Converged FiWi Access Network," *IEEE Transactions on Network and Service Management*, vol. 14, no. 3, pp. 498-513, September 2017.
- [66] A. Kapsalis, P. Kasnesis, I. S. Venieris, D. I. Kaklamani, and C. Z. Patrikakis, "A Cooperative Fog Approach for Effective Workload Balancing," *IEEE Cloud Computing*, vol. 4, no. 2, pp. 36-45, March 2017.
- [67] D. Bernstein, "Containers and Cloud: From LXC to Docker to Kubernetes," *IEEE Cloud Computing*, vol. 1, no. 3, pp. 81-84, September 2014.
- [68] D. Zhao, M. Mohamed, and H. Ludwig, "Locality-aware Scheduling for Containers in Cloud Computing," *IEEE Transactions on Cloud Computing*, January 2018, available online.
- [69] D. Towsley, "Mobility Models for Wireless Networks: Challenges, Pitfalls, and Successes," in *Proceedings of the 2008 22nd Workshop on Principles of Advanced and Distributed Simulation*, June 2008, pp. 3-3.
- [70] J. Plachy, Z. Becvar, and C. Strinati, "Dynamic Resource Allocation Exploiting Mobility Prediction in Mobile Edge Computing," in *Proceedings of the 2016 IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, September 2016, pp. 1-6.
- [71] A. Al-Shuwaili and O. Simeone, "Energy-Efficient Resource Allocation for Mobile Edge Computing-Based Augmented Reality Applications," *IEEE Wireless Communications Letters*, vol. 6, no. 3, pp. 398-401, June 2017.

- [72] A. Kiani and N. Ansari, “Toward Hierarchical Mobile Edge Computing: An Auction-Based Profit Maximization Approach,” *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 2082-2091, December 2017.
- [73] X. Sun and N. Ansari, “Latency Aware Workload Offloading in the Cloudlet Network,” *IEEE Communications Letters*, vol. 21, no. 7, pp. 1481-1484, July 2017.
- [74] X. Chen, W. Li, S. Lu, Z. Zhou, and X. Fu, “Efficient Resource Allocation for On-Demand Mobile-Edge Cloud Computing,” *IEEE Transactions on Vehicular Technology*, vol. 67, no. 9, pp. 8769-8780, October 2018.
- [75] C. Vicentini, A. Santin, E. Viegas, and V. Abreu, “A Machine Learning Auditing Model for Detection of Multi-Tenancy Issues Within Tenant Domain,” in *Proceedings of the 2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, May 2018, pp. 543-552.
- [76] 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 IEEE 84th Vehicular Technology Conference (VTC-Fall)*, September 2016, pp. 1-5.
- [77] R. Tandon and O. Simeone, “Harnessing Cloud and Edge Synergies: Toward an Information Theory of Fog Radio Access Networks,” *IEEE Communications Magazine*, vol. 54, no. 8, pp. 44-50, August 2016.
- [78] J. Zhang, W. Xia, Z. Cheng, Q. Zou, B. Huang, F. Shen, F. Yan, and L. Shen, “An Evolutionary Game for Joint Wireless and Cloud Resource Allocation in Mobile Edge Computing,” in *Proceedings of the 2017 9th International Conference on Wireless Communications and Signal Processing (WCSP)*, October 2017, pp. 1-6.
- [79] C. Wang, F. R. Yu, C. Liang, Q. Chen, and L. Tang, “Joint Computation Offloading and Interference Management in Wireless Cellular Networks

- with Mobile Edge Computing,” *IEEE Transactions on Vehicular Technology*, vol. 66, no. 8, pp. 7432-7445, August 2017.
- [80] D. Ganguly, M. H. Mofrad, T. Znati, R. Melhem, and J. R. Lange, “Harvesting Underutilized Resources to Improve Responsiveness and Tolerance to Crash and Silent Faults for Data-Intensive Applications,” in *Proceedings of the 2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*, June 2017, pp. 536-543.
- [81] X. Lyu, W. Ni, H. Tian, R. P. Liu, X. Wang, G. B. Giannakis, and A. Paulraj, “Optimal Schedule of Mobile Edge Computing for Internet of Things Using Partial Information,” *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2606-2615, November 2017.
- [82] A. Crutcher, C. Koch, K. Coleman, J. Patman, F. Esposito, and P. Calyam, “Hyperprofile-Based Computation Offloading for Mobile Edge Networks,” in *Proceedings of the 2017 IEEE 14th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, October 2017, pp. 525-529.
- [83] K. Han, S. Li, S. Tang, H. Huang, S. Zhao, G. Fu, and Z. Zhu, “Application-Driven End-to-End Slicing: When Wireless Network Virtualization Orchestrates With NFV-Based Mobile Edge Computing,” *IEEE Access*, vol. 6, pp. 26 567-26 577, May 2018.
- [84] B. Li, K. Wang, D. Xue, and Y. Pei, “K-Means Based Edge Server Deployment Algorithm for Edge Computing Environments,” in *Proceedings of the 2018 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computing, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCOM/IOP/SCI)*, October 2018, pp. 1169-1174.
- [85] Y. Li and S. Wang, “An Energy-Aware Edge Server Placement Algorithm in Mobile Edge Computing,” in *Proceedings of the 2018 IEEE International Conference on Edge Computing (EDGE)*, July 2018, pp. 66-73.

- [86] K. Xiao, Z. Gao, Q. Wang, and Y. Yang, “A Heuristic Algorithm Based on Resource Requirements Forecasting for Server Placement in Edge Computing,” in *Proceedings of the 2018 IEEE/ACM Symposium on Edge Computing (SEC)*, October 2018, pp. 354-355.
- [87] M. Marjanovic, A. Antonic, and I. P. Zarko, “Edge Computing Architecture for Mobile Crowdsensing,” *IEEE Access*, vol. 6, pp. 10662-10674, January 2018.
- [88] L. Tianze, W. Muqing, Z. Min, and L. Wenxing, “An Overhead-Optimizing Task Scheduling Strategy for Ad-hoc Based Mobile Edge Computing,” *IEEE Access*, vol. 5, pp. 5609-5622, March 2017.
- [89] Y. Sun, S. Zhou, and J. Xu, “EMM: Energy-Aware Mobility Management for Mobile Edge Computing in Ultra Dense Networks,” *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2637-2646, November 2017.
- [90] S. Wang, J. Xu, N. Zhang, and Y. Liu, “A Survey on Service Migration in Mobile Edge Computing,” *IEEE Access*, vol. 6, pp. 23 511-23 528, April 2018.
- [91] H. Trinh, P. Calyam, D. Chemodanov, S. Yao, Q. Lei, F. Gao, and K. Palaniappan, “Energy-Aware Mobile Edge Computing and Routing for Low-Latency Visual Data Processing,” *IEEE Transactions on Multimedia*, vol. 20, no. 10, pp. 2562-2577, October 2018.
- [92] H. Zhang, Z. Chen, J. Wu, and K. Liu, “FRRF: A Fuzzy Reasoning Routing-Forwarding Algorithm Using Mobile Device Similarity in Mobile Edge Computing-Based Opportunistic Mobile Social Networks,” *IEEE Access*, vol. 7, pp. 35874-35889, March 2019.
- [93] 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.

- [94] 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.
- [95] 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.
- [96] C. E. Shannon, “A Mathematical Theory of Communication,” *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379-423, 1948.
- [97] K. McClaning and T. Vito, *Radio Receiver Design*. Noble Publishing Corporation, 2000.
- [98] B. Sklar, “Rayleigh Fading Channels in Mobile Digital Communication Systems. I. Characterization,” *IEEE Communications Magazine*, vol. 35, no. 7, pp. 90-100, July 1997.
- [99] G. Miao, J. Zander, K. W. Sung, and S. B. Slimane, *Fundamentals of Mobile Data Networks*. Cambridge University Press, 2016.
- [100] I. Adan and J. Resing, *Queueing Theory*. Eindhoven University of Technology Eindhoven, 2002.
- [101] 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.
- [102] 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.
- [103] S. Rathore, P. K. Sharma, A. K. Sangaiah, and J. J. Park, “A Hesitant Fuzzy Based Security Approach for Fog and Mobile-Edge Computing,” *IEEE Access*, vol. 6, pp. 688-701, January 2018.

- [104] X. He, J. Liu, R. Jin, and H. Dai, “Privacy-Aware Offloading in Mobile-Edge Computing,” in *Proceedings of the 2017 IEEE Global Communications Conference (GLOBECOM 2017)*, December 2017, pp. 1-6.
- [105] S. Matoussi, I. Fajjari, S. Costanzo, N. Aitsaadi, and R. Langar, “A User Centric Virtual Network Function Orchestration for Agile 5G Cloud-RAN,” in *Proceedings of the 2018 IEEE International Conference on Communications (ICC)*, May 2018, pp. 1-7.
- [106] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, “A Survey on Mobile Edge Computing: The Communication Perspective,” *IEEE Communications Surveys Tutorials*, vol. 19, no. 4, pp. 2322-2358, November 2017.
- [107] H. Wen, L. Yang, and Z. Wang, “ParGen: A Parallel Method for Partitioning Data Stream Applications in Mobile Edge Computing,” *IEEE Access*, vol. 6, pp. 5037-5048, January 2018.
- [108] D. Li, B. Dong, E. Wang, and M. Zhu, “A Study on Flat and Hierarchical System Deployment for Edge Computing,” in *Proceedings of the 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, January 2019, pp. 163-169.
- [109] J. Kennedy, “Particle Swarm Optimization,” in *Encyclopedia of Machine Learning*. Springer, 2011, pp. 760-766.
- [110] 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.
- [111] 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.

- [112] 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.
- [113] 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.
- [114] Y. Wang, X. Lin, and M. Pedram, “A Nested Two Stage Game-Based Optimization Framework in Mobile Cloud Computing System,” in *Proceedings of the 2013 IEEE 7th International Symposium on Service Oriented System Engineering (SOSE)*, March 2013, pp. 494-502.
- [115] D. S. Abdelminaam, H. M. A. Kader, M. M. Hadhoud, and S. M. El-Sayed, “Elastic Framework for Augmenting the Performance of Mobile Applications using Cloud Computing,” in *Proceedings of the 2013 9th International Computer Engineering Conference (ICENCO)*, December 2013, pp. 134-141.
- [116] 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.
- [117] 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.
- [118] 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.
- [119] 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.

- [120] 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.
- [121] 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.
- [122] 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.
- [123] 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.
- [124] J. Armstrong, “OFDM for Optical Communications,” *Journal of Lightwave Technology*, vol. 27, no. 3, pp. 189- 204, February 2009.
- [125] 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.
- [126] L. W. Barclay, *Propagation of Radiowaves, 2nd Edition*. Institution of Engineering and Technology, April 2003.
- [127] L. Jiang, R. Wang, J. Yuan, H. Luo, L. Yu, and Y. Duan, “Optimization Design of Electromagnetic Relay Based on Improved Particle Swarm Optimization Algorithm,” in *Proceedings of the 2018 2nd IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, May 2018, pp. 202-205.

- [128] K. Xing, C. Hu, J. Yu, X. Cheng, and F. Zhang, “Mutual Privacy Preserving k-Means Clustering in Social Participatory Sensing,” *IEEE Transactions on Industrial Informatics*, vol. 13, no. 4, pp. 2066-2076, August 2017.
- [129] S. J. Nawaz, S. K. Sharma, S. Wyne, M. N. Patwary, and M. Asaduzzaman, “Quantum Machine Learning for 6G Communication Networks: State-of-the-Art and Vision for the Future,” *IEEE Access*, vol. 7, pp. 46 317-46 350, April 2019.
- [130] G. R. MacCartney, J. Zhang, S. Nie, and T. S. Rappaport, “Path Loss Models for 5G Millimeter Wave Propagation Channels in Urban Micro-cells,” in *Proceedings of the 2013 IEEE Global Communications Conference (GLOBECOM)*, December 2013, pp. 3948-3953.
- [131] 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.
- [132] 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.
- [133] 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.
- [134] 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.
- [135] 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.

- [136] 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.
- [137] 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.
- [138] 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.
- [139] 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.
- [140] 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.
- [141] 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.

# Publications

## Journals

- [1] Tiago Koketsu Rodrigues, Katsuya Suto and Nei Kato, "Edge Cloud Server Deployment with Transmission Power Control through Machine Learning for 6G Internet of Things," in IEEE Transactions on Emerging Topics in Computing, available online.
- [2] Tiago Koketsu Rodrigues, Katsuya Suto, Hiroki Nishiyama, Jiajia Liu and Nei Kato, "Machine Learning meets Computation and Communication Control in Evolving Edge and Cloud: Challenges and Future Perspective," in IEEE Communications Surveys and Tutorials, available online.
- [3] Keisuke Miyanabe, Tiago Gama Rodrigues, Yunseong Lee, Hiroki Nishiyama and Nei Kato, "An Internet of Things Traffic-Based Power Saving Scheme in Cloud-Radio Access Network," in IEEE Internet of Things Journal, vol. 6, no. 2, pp. 3087-3096, April 2019.
- [4] Hiroki Nishiyama, Tiago Gama Rodrigues and Jiajia Liu, "A Probabilistic Approach to Deploying Disaster Response Network," in IEEE Transactions on Vehicular Technology, vol. 67, no. 12, pp. 12086-12094, December 2018.
- [5] Tiago Gama Rodrigues, Katsuya Suto, Hiroki Nishiyama, Nei Kato and Katsuhiro Temma, "Cloudlets Activation Scheme for Scalable Mobile Edge Computing with Transmission Power Control and Virtual Machine Migration," in IEEE Transactions on Computers, vol. 67, no. 9, pp. 1287-1300, September 2018. (IEEE Transactions on Computers Sempter 2018 Spot-

light Paper) (2018 Best Paper Award from IEEE Transactions on Computers)

- [6] Tiago Gama Rodrigues, Katsuya Suto, Hiroki Nishiyama and Nei Kato, "Hybrid Method for Minimizing Service Delay in Edge Cloud Computing Through VM Migration and Transmission Power Control," in IEEE Transactions on Computers, vol. 66, no. 5, pp. 810-819, May 2017.

## Refereed Conference Papers

- [7] Tiago Koketsu Rodrigues, Katsuya Suto and Nei Kato, "Hyperparameter Study of Machine Learning Solutions for the Edge Server Deployment Problem," 2019 IEEE 90th Vehicular Technology Conference (VTC-Fall), Honolulu, HI, USA, September 2019, accepted.
- [8] Tiago Gama Rodrigues, Katsuya Suto, Hiroki Nishiyama and Nei Kato, "A PSO Model with VM Migration and Transmission Power Control for Low Service Delay in the Multiple Cloudlets ECC Scenario," 2017 IEEE International Conference on Communications (ICC), Paris, France, May 2017.
- [9] Katsuya Suto, Tiago Gama Rodrigues, Hiroki Nishiyama, Nei Kato, Hirotaka Ujikawa and Ken-Ichi Suzuki, "QoE-Guaranteed and Sustainable User Position Guidance for Post-Disaster Cloud Radio Access Network," 2016 IEEE Global Communications Conference (GLOBECOM), Washington, DC, USA, December 2016.
- [10] Tiago Gama Rodrigues, Katsuya Suto, Hiroki Nishiyama, Nei Kato, Kimihiro Mizutani, Takeru Inoue and Osamu Akashi, "Towards a Low-Delay Edge Cloud Computing through a Combined Communication and Computation Approach," 2016 IEEE 84th Vehicular Technology Conference (VTC-Fall), Montreal, QC, Canada, September 2016. (IEEE VTS Tokyo Chapter 2016 Young Researcher 's Encouragement Award)

## **Non-Refereed Conference Papers**

- [11] Tiago Gama Rodrigues, "A Study on Improving Service Delay and Sustainability in Edge Cloud Computing," Annual Workshop on A3 Foresight Program, Pyeongchang, Korea, July 2016. (Best Presentation Award)

## **Domestic Conference Papers**

- [12] Tiago Gama Rodrigues, Katsuya Suto, Hiroki Nishiyama and Nei Kato, "Cross-Acceleration Study of Particle Swarm Optimization Applied to Resource Minimization in Mobile Edge Computing," 2018 IEICE General Conference, Tokyo Denki University, Tokyo, Japan, March 2018.
- [13] Tiago Gama Rodrigues, Katsuya Suto, Hiroki Nishiyama and Nei Kato, "Parameter Analysis for the Application of Particle Swarm Optimization to the Multiple Cloudlets Edge Cloud Computing Problem," 2017 IEICE Society Conference, Tokyo City University, Tokyo, Japan, September 2017.

## **Awards**

- [14] A3 Workshop 2016 Best Presentation Award - "A Study on Improving Service Delay and Sustainability in Edge Cloud Computing"
- [15] IEEE VTS Tokyo Chapter 2016 Young Researcher's Encouragement Award - "Towards a Low-Delay Edge Cloud Computing through a Combined Communication and Computation Approach"
- [16] Tohoku University Graduate School of Information Sciences Dean's Award 2017
- [17] 2017 IEEE ComSoc Sendai Chapter Student Excellent Research Award
- [18] 2018 Tohoku University Graduate School of Information Sciences PhD Course Midterm Best Presentation

- [19] IEEE Transactions on Computers Sempter 2018 Spotlight Paper - "Cloudlets Activation Scheme for Scalable Mobile Edge Computing with Transmission Power Control and Virtual Machine Migration"
- [20] 2018 Best Paper Award from IEEE Transactions on Computers - "Cloudlets Activation Scheme for Scalable Mobile Edge Computing with Transmission Power Control and Virtual Machine Migration"



# Appendix

## Copyright Permissions

In this appendix, we include the permissions that were used to write this thesis. Please see the attached documents for a detailed description of permissions. The used publications that were used to write this thesis are listed as follows:

- T. K. Rodrigues, K. Suto and N. Kato, "MEdge Cloud Server Deployment with Transmission Power Control through Machine Learning for 6G Internet of Things," in *IEEE Transactions on Emerging Topics in Computing*. Available online.
- T. K. Rodrigues, K. Suto, H. Nishiyama, J. Liu and N. Kato, "Machine Learning meets Computation and Communication Control in Evolving Edge and Cloud: Challenges and Future Perspective," in *IEEE Communications Surveys & Tutorials*. Available online.
- T. G. Rodrigues, K. Suto, H. Nishiyama, N. Kato and K. Temma, "Cloudlets Activation Scheme for Scalable Mobile Edge Computing with Transmission Power Control and Virtual Machine Migration," in *IEEE Transactions on Computers*, vol. 67, no. 9, pp. 1287-1300, September 2018.
- 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," in *IEEE Transactions on Computers*, vol. 66, no. 5, pp. 810-819, May 2017.
- 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)*, Paris, pp. 1-6, May 2017.

In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity’s name goes here]’s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for re-sale or redistribution, please go to [http://www.ieee.org/publications\\_standards/publications/rights/rights\\_link.html](http://www.ieee.org/publications_standards/publications/rights/rights_link.html) to learn how to obtain a License from RightsLink.