2018 IEEE 6th International Conference on Future Internet of Things and Cloud

A Rank Scheduling Mechanism for Fog environments

David Perez Abreu*, Karima Velasquez*, Marcio R. M. Assis†,
Luiz Bittencourt†, Marilia Curado*, Edmundo Monteiro*, and Edmundo Madeira†
*CISUC, Department of Informatics Engineering
University of Coimbra, Portugal
Email: {dabreu,kcastro,marilia,edmundo}@dei.uc.pt
†Institute of Computing
University of Campinas, Brazil
Email: {marcio.assis,bit,edmundo}@ic.unicamp.br

***Abstract*—With the advent of the Internet of Things many applications emerged that are not suitable for well-known paradigms like the Cloud, requiring its extension to provide more features to final users. Thus, the Fog rises as an extension to the Cloud able to provide mobility support, geographical distribution, and lower latency, by moving the services closer to the users, to the edge of the network. This new environment located at the edge comes with its own orchestration challenges. Among the orchestration functions that must be adapted to this new environment is scheduling. This paper presents a simple scheduling algorithm for Fog federative environments that organizes Fog instances into divisions for task assignment. Experimental results show that this approach could be particularly beneficial for critical time applications, commonly located at the Fog.**

***Index Terms*—Scheduling; Fog; Federative Fog; Ranking.**

I. Introduction

The need to meet the requirements of a new and emerging range of applications motivated the extension of the Cloud computing paradigm. These new applications are characterized by being present in end user devices (e.g., smartphones, wearable devices) and embedded in the Internet of Things (IoT) paradigm (e.g., sensors and actuators). Having as main requirements the geographic dispersion and the execution in real-time [1], these applications have defied the until then existing paradigms (i.e., Cloud computing). To tackle the geographic dispersion, within other motivations, came the interconnected Clouds, or simply Interclouds or Cloud federations. Organized in associations, several geographically dispersed Cloud Service Providers (CSPs) cooperated with each other by offering idle resources and consumed resources under more attractive conditions when needed [2]. However, even Interclouds were not able to run this new class of applications in real-time. This is because, the distance of the applications present in wearable and other IoT devices remain far from the CSPs [3], [4], which adds latency to the process.

To overcome the Cloud limitation previously described, data storage, data processing, and network activity were brought to the edge of the network, close to end users. This derivation of the Cloud computing paradigm is called Fog computing [5], [6]. Fog is an environment with a collection of heterogeneous devices that work in a decentralized way, communicating and cooperating symbiotically [7], [8]. Among the properties that Fog computing seeks to deliver are: *high resilience*, *low latency*, *choreography management*, and *scalability* [9], [10], [11]. Additionally, Fog computing deals with some elements inherent to the network edge, such as *location awareness* and *mobility support* [6], [1].

The Fog characteristics previously discussed enable and require scheduling mechanisms, able to take application demands into account when placing tasks in the Fog nodes. The scheduling problem can be modeled in the Fog taking into consideration applications and requirements, such as latency and mobility. Currently, many scheduling algorithms focus on Cloud computing, but scheduling for Fog computing requires some adaptations to take into consideration the differences in the characteristics between both environments.

The heterogeneity of devices present in Fog and the characteristics of the applications executed under them highlight the importance of developing new approaches toward orchestration, and in particular scheduling. Modeling of the orchestration components that need to be able to perform the deployment of tasks and handle them inside the environment involves many challenges. Using the well-known model for network management functions Fault, Configuration, Accounting, Performance, and Security (FCAPS) [12] and more recent efforts to extrapolate it to distributed and virtualized environments [13], [14], it is possible to identify the main challenges that a Fog orchestrator must handle (see Fig. 1).

Orchestration in Fog environments is a complex function composed of many sub-functions. One of said sub-functions is the scheduling of tasks [11]. In this paper, we propose a simple scheduling algorithm for task scheduling in an inter Fog environment, or federative Fog environment, where Fog Service Providers (FPSs) could capitalize in the cooperation between Fog instances, or *Cloudlets*, while also achieving load balancing among the Fog nodes.

Experimental results show that the proposed scheduling mechanism outperforms more classical approaches in some cases that could be particularly beneficial for critical-time applications, which usually reside in the Fog.

978-1-5386-7503-8/18/$31.00 ©2018 IEEE
DOI 10.1109/FiCloud.2018.00059

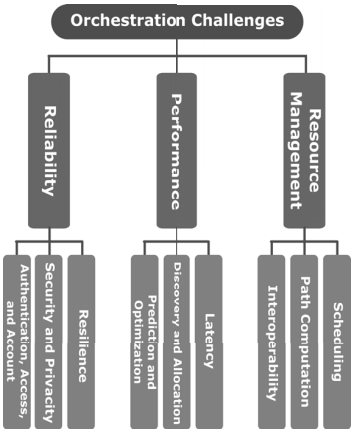

Fig. 1. Main challenges faced by orchestrators in Fog computing environments.

This paper is organized as follows. Section II shows a revision of the state of the art. Section III describes the proposed scheduling mechanism, based on a Ranking approach. Section IV details the validation of our proposal, via simulations. Section V presents some discussions about challenges and possible future research directions. Section VI concludes the paper.

II. Related Work

The topic of scheduling in computing system has vastly been explored. Some of the traditional mechanisms are First Come, First Served (FCFS), Round Robin (RR), Priority-based, among others [15]. In the case of Cloud computing, scheduling has also been thoroughly explored. Xu et al. [16] propose MQMW, a Multiple Quality of Service (QoS) constrained scheduling strategy for Multi-Workflows. The strategy enables different scheduling workflows taking into consideration QoS requirements. The proposed system consists of three elements: preprocessor, scheduler, and executor. Some attributes of the tasks are computed by the preprocessor and sent to the scheduler to prepare the queues by recomputing some attributes and sorting the tasks, and finally the executor picks the task from the queue. There is constant event-triggered communication between these elements, which could incur in network and processing overhead.

Li et al. [17] implement a scheduling policy based on Load Balancing Ant Colony Optimization (LBACO) algorithm for Cloud tasks, with the goal of achieving load balancing in the system. They use an evolutionary approach that selects the new task to schedule depending on the result of the past task. As it is usual in solutions based on evolutionary approaches, the time complexity is high. Furthermore, authors make assumptions that oversimplify the simulated environment by limiting the type of tasks to computationally intensive and defining that they are mutually independent.

Ningning et al. [18] present a solution where the physical nodes are turned into Virtual Machine (VM) nodes. They combine graph theory with the Fog computing paradigm to use graph partitioning mechanisms for scheduling. Creating, updating, and partitioning graphs in real-time could be time consuming, especially as the topology grows.

Ettikyala and Latha [19] propose a rank-based scheduler. The VMs that will host the tasks are sorted by their computational capacities and placed on a list; also the tasks are also sorted by their length. Thus, heavier tasks will be assigned to more powerful VMs. This approach requires an additional delay for the arriving tasks in order to be able to sort them out.

Pham and Huh [20] introduce a heuristic-based algorithm that performs a trade-off between execution time and the cost of using some Cloud resources. A Directed Acyclic Graph (DAG) represents the tasks, and another graph represents the nodes in the topology. The scheduling method first determines the task priority (based on some attributes such as computation time of task) and then selects the node where the task will be executed. The algorithm also considers the monetary cost for using Cloud resources (e.g., computing, bandwidth). Once again, including so many variables in the scheduling process, although refining the solution to a more sophisticated result, could delay the deployment of critical tasks.

Cardellini et al. [21] propose a QoS aware scheduler for Fog environment based on Apache Storm. The scheduler focused on network latency and traffic generated between nodes to allow self-adaptation capabilities to the system. Similarly, Zeng et al. [22] discuss an efficient task scheduling and resource management strategy for a specific type of application in the Fog.

The analyzed work covers different approaches towards scheduling in Cloud and Fog environments. However, for critical response time applications, they could be time-consuming thus delaying the final deployment of these critical tasks, especially in Fog environments where the devices usually are resource-constrained. In the next section, we describe a scheduling mechanism focused on ranking tasks and *Cloudlets* in Fog environments accordingly with their latency.

III. The Ranking orchestration approach

Fog computing environments are chiefly susceptible to latency considering the characteristics of the applications that run in this domain [23], [11]. Support of different latency requirements in the Fog directly impacts how the orchestration process in this environment must be done. Thus, it is necessary to consider different aspects during the orchestration process, such as the number of *Cloudlets* and their characteristics to improve the response time offered and fulfill applications' requirements. Additionally, this process should be performed in an efficient way to minimize the impact of the orchestration tasks in applications.

To address the orchestration challenges discussed so far, we proposed a Ranking orchestration mechanism focused on a Cloud-framing system to select the best *Cloudlet* that meets the applications' requirements. The main components of the Ranking orchestration are summarized in Table I and described below:

- Cloudlets: the set of all *Cloudlets*, denoted as C, present in the environment. A *Cloudlet* ($c \in C$) is an autonomous Fog instance domain in the infrastructure. *Cloudlets* are typically used to offload applications when devices are not capable of executing them.
- Divisions: represented by the set D, each *division* d_n, ($d_n \in D, n = 1, 2, \ldots$) is a logical partition C' of the *Cloudlets* present in the environment. *Divisions* are partially ordered, where the sorting criterion is defined by the property of interest (e.g., latency, resilience). There will always be at least two *divisions*: access division (d_n) and premier division (d_1). Specifically, d_n is the *division* where the new *Cloudlets* are inserted, and the premier division is where the *Cloudlets* with the best *scores* are grouped. Additionally, each *division* has associated a lower (t_{ini}) and upper (t_{end}) bound, these values define the thresholds to be considered to promote or demote *Cloudlets* by the n *divisions*.
- Score: the *scores* (s) are indexes used in the decision of promoting or demoting the ranking of each *Cloudlet* at the end of each *round*. The score defines to which *division* each *Cloudlet* belongs in a specific time-frame. These indexes can be obtained directly from the property of interest or can be calculated through samples obtained over time (e.g., latency measurements).
- Rounds: defined as predetermined time periods, each *round* r represents the static and consecutive intervals between two calculations of *score*. A *round* can be asynchronous or synchronous by *divisions* – they can have or not the same value for all *division* – and can be individual by *Cloudlet* or global by *divisions*. At the end of each *round* r the *scores* of the *Cloudlets* are determined and the process of promotion or demotion is executed.

TABLE I
NOTATION USED IN RANKING ORCHESTRATION MECHANISM.

NOTATION	
C	Set of all Cloudlets present in the environment.
c	Cloudlet which represents a Fog domain.
D	Set of all the divisions present in the environment.
d_n	n_{th} division, also denotes the access division.
d_1	Premier division.
t_{ini_n}	Lower threshold of division n.
t_{end_n}	Upper threshold of division n.
r	Period of time between two calculations of scores.
s	Index that determines the classification of each Cloudlet c.

The Ranking orchestration approach organizes the *Cloudlets* available in the environment (defined as the set C) into n *divisions*. Each $c \in C$ is allocated in a *division* d_n at a given time. This process is monitored by the *Referee*. To regular intervals of time (i.e., *rounds* r) a reclassification of each c is performed to determine the members of the n *divisions*. The reclassification is based on an index called *score* that is calculated from specific properties collected from each *Cloudlet* c (e.g., latency, resilience). After each *round* r, a *Cloudlet* c can be promoted or demoted by the n *divisions* depending on its *score*. Using the Ranking approach, the search space to find where to host an application is reduced considering the grouping of *Cloudlets* into n *divisions*; as a result, the scheduler will search the subset $C' \subset C$ in order to choose a $c \in C'$ that belongs to the proper *division* which fulfills the application's requirements. Fig. 2 depicts the Ranking scheduling architecture in a Fog computing environment.

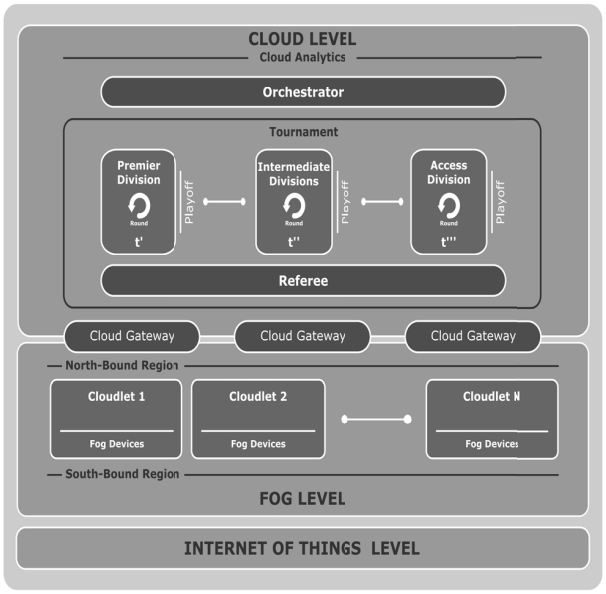

Fig. 2. Rank scheduling architecture in a Cloud/Fog environment.

A. The Ranking mechanism operation

The Ranking scheduling approach initially performs a *bootstrap* to stabilize the environment and define some important values, such as the number of *divisions* (the size of the set D) and their thresholds. Once this step is done, all the *Cloudlets* already available in the environment, as well as the new ones that arrive later on, are associated to the access *division* (d_n). If the *round* r used is global by *division*, it is necessary to set an initial *score* s to new *Cloudlets*. This initial *score* is required to avoid penalizing the *Cloudlets* that arrive just after a *round* has ended, whom would have to wait until the next *round* for a *score* update.

Upon receiving an application, the scheduler verifies its requirements and using the lower (t_{ini}) and upper (t_{end}) thresholds of each *division* (d_x), the scheduler selects a *Cloudlet* c from the *division* that best meets the application requirements to host it. Note that the term "best meets" considers only the resources in the *divisions* that minimally meets the application's requirements. For example, consider a scenario where there are three *divisions* where latency (in seconds) is used as the classification criteria. These *divisions* are segmented by thresholds: $1.0s$ – access division (d_n) – $0.67s$, $0.66s$ - intermediate division (d_2) – $0.34s$, $0.33s$ – premier division (d_1) – $0.0s$. If an application requires a

latency of $0.5s$, it will be serviced by the intermediate division rather than by the premier division, because the intermediate division minimally meets the application's requirements.

At the end of each *round* each *division* performs the calculation of the *score* based on the samples of interest, obtained from the *Cloudlets* that belong to the *division*. In order to guarantee the minimum requirements offered by each *division*, those *Cloudlets* which *scores* are below than the lower (t_{ini}) value of the *division* are marked for demotion, on the contrary, the *Cloudlets* which *scores* are above than the upper (t_{end}) of the *division* are marked for promotion.

B. The Ranking scope

The Ranking approach proposed could be applied in different scenarios and configured to respond to diverse interests. A typical scenario is related to how to schedule the tasks inside a single Cloud or Fog; however, the Rank mechanism could help to coordinate the interaction between *Cloudlets* in a Cloud or Fog federation.

The federation approach on Cloud or Fog lays in seamless and transparent access by the Cloud/Fog clients to Cloud/Fog services following a Service Level Agreement (SLA) [2]. In this scenario, the Ranking approach could be used to orchestrate a federation of Fog instances, for example, around their latency. Precisely, the Rank mechanism guarantees that the SLA, based on the latency offered by the *Cloudlet* that belongs to a Federation, will be met. Thus the *Cloudlets* are organized in *divisions* according to a particular property of interest (e.g., their latency), and the applications or tasks generated in a specific *Cloudlet* could only be executed in the *Cloudlets* that are in the same *division* where the former one belongs, or in a lower *division*.

In the next section, we describe the experiments performed to validate the Ranking scheduling approach, as well as, the scenario chosen and the assumptions that we made.

IV. Simulations and results

In order to validate the Ranking mechanism proposed and measure its performance, simulations were carried out. The simulator used was CloudSim Plus [24]. For comparison purposes, the classic scheduling algorithm of Round Robin was also implemented. For the experiments we used *Cloudlets* as the representation of a Micro Data Center (MDC) which has associated a set of ten physical hosts with the computational resource characteristics listed in Table II. Similarly, a variable amount of VMs were instantiated (one VM per each group of four tasks) with the features described in the same table (see Table II). The VMs are destroyed when the last task of that group of four is completed.

TABLE II
Nodes' characteristics.

Node	# CPUs	CPU (MIPS)	RAM (MB)	Bandwidth (Mbps)	Storage (MB)
Hosts	4	1000	2048	10000	1000000
VMs	2	1000	512	10000	10000

In order to measure the performance of both approaches (Round Robin vs. Rank), the average latency of a server to complete a task is used as a metric. A lower average latency reflects a more evenly distributed set of tasks, while also has repercussions on the Quality of Experience (QoE) for the final users.

A detailed description of the scenarios designed for the evaluation, and the results obtained during the simulation process, are provided in the following subsections.

A. Scenario description

The scenario considered emulates a Fog federative environment where the Rank approach was used to coordinate the interaction between the Fog instances or *Cloudlets* that are participating in said federation. A *Cloudlet* c wants to join the federation in order to increase its capacities by sharing its workload. Furthermore, each *Cloudlet* has a *Broker* associated which is in charge of attending the application requests (i.e., *tasks*) submitted by the clients subscribed to the *Cloudlet* (*Broker*$_x$ attends the application requests submitted to c_x). Different amounts of *Cloudlets* were used on the simulation, this is from two to nine Fog instances.

When a new task arrives, the *Broker* decides in which c a new VM should be instantiated to run it. The selection process follows the Round Robin or Rank approaches, respectively. In the Rank approach, three *divisions* were implemented $n = 3$ (d_3, d_2, d_1), considering *latency* as the property of interest. Each *division* has dynamically calculated thresholds. Three *rounds* were defined (r', r'', r'''), one for each *division*, with they being asynchronous among them and global by *division*. To calculate the *scores* of each *division*, the average of the latency samples obtained from each *task* executed in the different *Cloudlets* was used. Due to the global aspect of the *rounds* (by *division*), it was necessary to set an initial *score* for each new *Cloudlet*. The *scores* were calculated by using an harmonized mean of all the participants in each *division*. Fig. 3 depicts the raking process of the *Cloudlets* during the simulation.

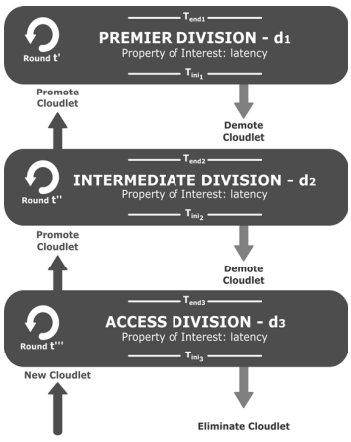

Fig. 3. Cloudlets' ranking process.

To determine the conditions where the Rank scheduling approach could be advantageous, several variations to the described scenario were simulated, changing different parameters. Three types of tasks were defined: small, medium, and big (see Table III). The proportion of each task type was changed between simulations, following a uniform distribution, in order to determine which approach is more suitable for each case (i.e., balanced, predominance of big tasks, predominance of medium tasks, predominance of small tasks,). On each case, tasks were randomly generated during each simulation until reaching 25000 requests.

TABLE III
TASK TYPE CHARACTERISTICS.

Task Type	# CPUs	Length (MIPS)	Input size (MB)	Output size (MB)
Small	1	1000 to 2665	1024	1024
Medium	1	2666 to 4334	1024	1024
Big	1	4335 to 6000	1024	1024

The other parameter that was changed was the amount of *Cloudlets* (the size of set C). This could enlight the selection of a convenient size to use for the Fog federation. Each scenario was simulated 30 times to reduce the statistical error. The results are reported in the following subsection.

B. Results

On the first experiment, depicted in Fig. 4, the different task type proportion is shown. The first case shows an even distribution between task types, the second shows more big task, the third more medium tasks, and the fourth and final one more small tasks. For this experiment, the amount of *Cloudlets* was fixed to five. On this case, for fairly distribution of task types (33%,33%,33%), the Rank and Round Robin approaches show similar mean latency values, however, the Rank approach shows a more compact boxplot, thus indicating a lower standard deviation. This could benefit jitter-sensitive applications, since the delays are more predictable. For the other task type distributions the Rank approach outperforms Round Robin, although only for a few milliseconds. This difference could be particularly significant for critical time applications, that are usually deployed in the Fog.

Fig. 5 shows the contrast of using different numbers of *Cloudlets*. In this experiment, the size of the set C was changed from two to nine, to determine the influence on the number of members of the federation in the latency. For this case, the task type proportion was fixed to balanced (33%, 33%, 33%). Since the amount and types of requests remained the same, the overall latency shows a steady value even by augmenting the number of *Cloudlets* (i.e., the size of C), around 3.5 seconds. It is noticeable that although the mean values are fairly similar, in most cases (around 63%) the Rank mechanism is slightly better than Round Robin.

The results obtained show that the Rank approach is more appropriate for some circumstances than others. A discussion is provided in the following section.

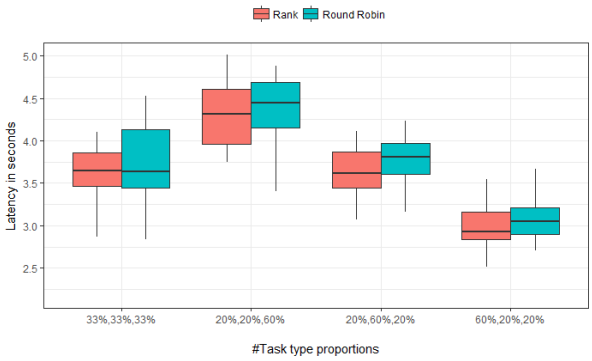

Fig. 4. Latency under different task type proportions.

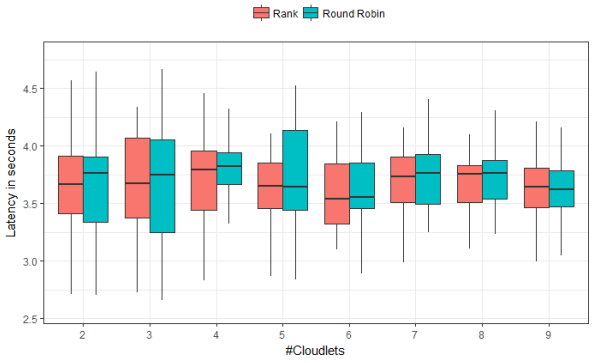

Fig. 5. Task latency changing the number of MDCs in the Fog federation.

V. DISCUSSION

The results obtained show that the Rank scheduling approach has a lower mean latency value for most of the evaluated cases; also the distribution of the values is more compact which means that the Rank mechanism has a more predictable behavior.

It is noteworthy that the difference in the mean latency value between both approaches is relatively small, around 100 milliseconds when using only two *Cloudlets*. The difference is reduced as the number of size of set C is increased. While for some type of applications this difference is negligible, for some particular applications deployed in Fog environments it could be significant. Applications such as robotics and telepresence, factory automation, health care, and virtual reality require latency levels in the order of milliseconds [25], [26].

Currently, with the advent of novel computing paradigms such as the IoT, these applications are being brought closer to the user using the Fog paradigm, where the simulated *Cloudlets* are located. Thus, a difference in the order of milliseconds could represent a major improvement for these types of applications, although probably not so symbolic for more conventional services.

The Rank approach results obtained in this research lead to consider its use for some particular Fog deployments. In Fog environments, it is usual to have several tiers (N-Tiers) of

nodes or groups of nodes that need to be orchestrated [26], for example, by their latency as feature of interest. Considering this niche, the Rank mechanism could offer a way to organize the interaction between the *Cloudlets*, as well as, help to determine the amount of tiers which could be represented as the *divisions* in our proposed solution.

Orchestration, and particularly scheduling, in Fog environments is a promising and challenging field of research. In the Fog it is required to deal with several metrics and properties that directly affect the QoS and the QoE of the applications and users. Thus, more efforts are needed in this area to better support application requirements in Fog environments. The Rank scheduling mechanism described above is an example of how a simple scheduling solution can improve the performance of applications in Fog environments.

VI. Conclusions

In this paper, we propose a simple scheduling algorithm for Fog federated environments. The proposal organizes the Fog instances, called *Cloudlets*, into *divisions* according to the mean latency of the Fog nodes to complete a task. *Cloudlets* could be promoted to higher *divisions*, or demoted to lower ones, according to the performance of their nodes during a period of time. The proposed algorithm was implemented and tested using simulations. For comparison purposes, the classic scheduling approach of Round Robin was used.

The Rank approach showed to have better results than Round Robin for most of the simulated scenarios. It is noteworthy however, that the difference in the results obtained by both approaches is in the order of milliseconds. While for some applications this could be negligible, for critical time applications this difference is more significant. Since these time-constrained applications are usually deployed at the Fog, this type of solutions for scheduling could prove to be substantially beneficial for the orchestration of the overall environment; moreover, given its simplicity, it would not imply an overload of computing nor communication among the nodes.

For future research we propose to expand the set of performed experiments by adding and removing *Cloudlets* in the federation during the simulation and measuring the impact of these actions; also varying the *Cloudlets*' characteristics. Furthermore, we plan to refine the scheduling algorithm by considering additional metrics for the ranking, such as energy consumption and resilience.

Acknowledgment

Karima Velasquez and David Perez Abreu wish to acknowledge the Portuguese funding institution FCT - Foundation for Science and Technology for supporting their research under the Ph.D. grants ≪SFRH/BD/119392/2016≫ and ≪SFRH/BD/117538/2016≫ respectively.

The work presented in this paper was partially carried out in the scope of the project SORTS, financed by the the CAPES - Coordenção de Aperfeiçoamento de Pessoal de Nível Superior ≪CAPES-FCT/8572/14-3≫ and by the FCT - Foundation for Science and Technology ≪FCT/13263/4/8/2015/S≫.

References

[1] L. F. Bittencourt, J. Diaz-Montes, R. Buyya, O. F. Rana, and M. Parashar, "Mobility-aware application scheduling in fog computing," *IEEE Cloud Computing*, vol. 4, no. 2, pp. 26–35, March 2017.

[2] H. Li, "Cloud federation as a service," Aug. 29 2017, US Patent 9,749,398.

[3] S. Yi, C. Li, and Q. Li, "A survey of fog computing: Concepts, applications and issues," in *Proceedings of the 2015 Workshop on Mobile Big Data*, ser. Mobidata '15. New York, NY, USA: ACM, 2015, pp. 37–42.

[4] S. Yi, Z. Hao, Z. Qin, and Q. Li, "Fog computing: Platform and applications," in *2015 Third IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb)*, Washington, DC, USA, Nov 2015, pp. 73–78.

[5] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, ser. MCC '12, 2012, pp. 13–16.

[6] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, *Fog Computing: A Platform for Internet of Things and Analytics*. Cham: Springer International Publishing, 2014, pp. 169–186.

[7] L. M. Vaquero and L. Rodero-Merino, "Finding your way in the fog: Towards a comprehensive definition of fog computing," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 5, pp. 27–32, Oct. 2014.

[8] P. Varshney and Y. Simmhan, "Demystifying fog computing: Characterizing architectures, applications and abstractions," in *2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC)*. Madrid, Spain: IEEE, May 2017, pp. 115–124.

[9] S. C. Hung, H. Hsu, S. Y. Lien, and K. C. Chen, "Architecture harmonization between cloud radio access networks and fog networks," *IEEE Access*, vol. 3, pp. 3019–3034, Dec 2015.

[10] M. Chiang and T. Zhang, "Fog and iot: An overview of research opportunities," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 854–864, Dec 2016.

[11] K. Velasquez, D. P. Abreu, D. Gonalves, L. Bittencourt, M. Curado, E. Monteiro, and E. Madeira, "Service orchestration in fog environments," in *2017 IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud)*, Prague, Czech Republic, Aug 2017, pp. 329–336.

[12] A. Clemm, *Network Management Fundamentals*, ser. Cisco Press, 2006.

[13] J. Yu and R. Buyya, "A taxonomy of workflow management systems for grid computing," *Journal of Grid Computing*, vol. 3, no. 3, pp. 171–200, Sep 2005. [Online]. Available: https://doi.org/10.1007/s10723-005-9010-8

[14] R. Mijumbi, J. Serrat, J. l. Gorricho, S. Latre, M. Charalambides, and D. Lopez, "Management and orchestration challenges in network functions virtualization," *IEEE Communications Magazine*, vol. 54, no. 1, pp. 98–105, Jan 2016.

[15] P. Salot, "A survey of various scheduling algorithm in cloud computing environment," *International Journal of Research in Engineering and Technology*, vol. 2, no. 2, pp. 131–135, 02 2013.

[16] M. Xu, L. Cui, H. Wang, and Y. Bi, "A multiple qos constrained scheduling strategy of multiple workflows for cloud computing," in *2009 IEEE International Symposium on Parallel and Distributed Processing with Applications*, Chengdu, China, Aug, pp. 629–634.

[17] K. Li, G. Xu, G. Zhao, Y. Dong, and D. Wang, "Cloud task scheduling based on load balancing ant colony optimization," in *2011 Sixth Annual Chinagrid Conference*, Liaoning, China, Aug 2011, pp. 3–9.

[18] S. Ningning, G. Chao, A. Xingshuo, and Z. Qiang, "Fog computing dynamic load balancing mechanism based on graph repartitioning," *China Communications*, vol. 13, no. 3, pp. 156–164, March 2016.

[19] K. Ettikyala and Y. V. Latha, "Rank based efficient task scheduler for cloud computing," in *2016 International Conference on Data Mining and Advanced Computing (SAPIENCE)*, Ernakulam, India, March 2016, pp. 343–346.

[20] X.-Q. Pham and E.-N. Huh, "Towards task scheduling in a cloud-fog computing system," in *2016 18th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, Kanazawa, Japan, Oct 2016, pp. 1–4.

[21] V. Cardellini, V. Grassi, F. L. Presti, and M. Nardelli, "On qos-aware scheduling of data stream applications over fog computing infrastructures," in *2015 IEEE Symposium on Computers and Communication (ISCC)*. Larnaca, Cyprus: IEEE, July 2015, pp. 271–276.

[22] D. Zeng, L. Gu, S. Guo, Z. Cheng, and S. Yu, "Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system," *IEEE Transactions on Computers*, vol. 65, no. 12, pp. 3702–3712, Dec 2016.

[23] K. Velasquez, D. Perez Abreu, M. Curado, and E. Monteiro, "Service placement for latency reduction in the internet of things," *Annals of Telecommunications*, vol. 72, pp. 105–115, 2017.

[24] M. C. S. Filho, R. L. Oliveira, C. C. Monteiro, P. R. M. Incio, and M. M. Freire, "Cloudsim plus: A cloud computing simulation framework pursuing software engineering principles for improved modularity, extensibility and correctness," in *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, Lisbon, Portugal, May 2017, pp. 400–406.

[25] I. Parvez, A. Rahmati, I. Güvenç, A. I. Sarwat, and H. Dai, "A survey on low latency towards 5g: Ran, core network and caching solutions," *CoRR*, vol. abs/1708.02562, 2017. [Online]. Available: http://arxiv.org/abs/1708.02562

[26] O. C. A. W. Group, "OpenFog Reference Architecture for Fog Computing," OpenFog Consortium, Tech. Rep., Feb 2017.