Self-organizing Energy-minimization Placement of QoE-constrained Services at the Edge*

Matteo Mordacchini[2][0000-0002-1406-828X], Luca Ferrucci[1][0000-0003-4159-0644], Emanuele Carlini[1][0000-0003-3643-5404], Hanna Kavalionak[1][0000-0002-8852-3062], Massimo Coppola[1][0000-0002-7937-4157], and Patrizio Dazzi[1][0000-0001-8504-1503]

[1] ISTI-CNR, Pisa, Italy
name.surname@isti.cnr.it
[2] IIT-CNR, Pisa, Italy
matteo.mordacchini@iit.cnr.it

Abstract. The wide availability of heterogeneous resources at the Edge of the network is gaining a central role in defining and developing new computing paradigms for both the infrastructures and the applications. However, it becomes challenging to optimize the system's behaviour, due to the Edge's highly distributed and dynamic nature. Recent solutions propose new decentralized, self-adaptive approaches to face the needs of this scenario. One of the most challenging aspect is related to the optimization of the system's energy consumption. In this paper, we propose a fully decentralized solution that limits the energy consumed by the system, without failing to match the users expectations, defined as the services' Quality of Experience (QoE.). Specifically, we propose a scheme where the autonomous coordination of entities at Edge is able to reduce the energy consumption by reducing the number of instances of the applications executed in system. This result is achieve without violating the services' QoE, expressed in terms of latency. Experimental evaluations through simulation conducted with PureEdgeSim demonstrate the effectiveness of the approach.

Keywords: Edge Computing · Self-organizing · Another keyword.

1 Introduction

Traditional Cloud solutions are facing increasing difficulties in coping with novel sets of applications, like latency-sensitive ones. The Edge/Cloud continuum paradigm allows to overcome these limits by seamlessly integrating one (or more) Cloud(s) and wide numbers of Edge resources, geographically distributed. However, several challenges are emerging for the management, coordination and optimization of these large sets of heterogeneous and dispersed resources [25]. Among

* This work has been partially supported by the European Union's Horizon 2020 Research and Innovation program, under the project ACCORDION (Grant agreement ID: 871793)

those challenges, a sensitive problem concerns the reduction of the overall energy consumption of the system. One way to achieve this results is to optimize the placement of the instances of the applications requested by the users. This is a non-trivial task, since it has to take into account the functional needs of the applications, the computational limits of Edge resources and the non-functional requirements associated with the users' Quality of Experience (QoE).

Distributed [1], self-organizing [20, 14, 10] and adaptive [3, 5] solutions have been advanced for facing these kind of challenges at the Edge. In this paper, we propose a decentralized, self-organizing and QoE-aware scheme for the optimization of the energy consumed by the system. Specifically, Edge entities interact among themselves and exchange information in order to determine whether the users of each application can be served using a lower number of instances. This behaviour allows to reduce the number of instances executed in the system, thus reducing the overall energy consumed. When taking the decision to shut down a potential redundant instance, the entities exploit the data they have exchanged to evaluate whether this decision is in accordance with the services' QoE and the computational limits of Edge resources. Experimental results through simulation show that the proposed solution is able to reduce the energy required by the system up to nearly 40%.

The rest of this paper is organized as follows. Section 2 contextualizes this work in the related scientific literature. Section 3 presents our the definition of the problem and the approach we propose. Section 4 describes the experimental evaluation of the proposed solution. Finally, Section 5 draws concluding remarks and highlights future work directions.

2 Related Work

Edge-based systems are the object of many investigations that try to optimize their performance by limiting the communications to/from centralized Clouds [23, 24]. In fact, these communications could introduce significant overhead and could potentially degrade the performance of many Edge-based applications, like locality and context-based services. A common way to overcome this problem is to use decentralized and/or self-organizing solutions [18, 11, 16]. These solutions achieve their goal by moving the applications [21, 15, 9] and/or data closer to users. When the data is moved in the system, the aim is to make it easy for the users to access it [22, 6, 19, 8]. In this case, the general strategy is to shorten the distance between the data storage devices or the data producers and their respective consumers [2, 13]. To achieve an optimization of the energy consumption levels of the entities at the Edge, we use a method which does not move data and/or applications closer to each other and/or closer to their users.

The optimization of the usage of Edge resources is proposed by Kavalionak et al. [12]. In this proposal, the devices fulfill their tasks by sharing and balancing the required computational costs. Beraldi et al. propose CooLoad [4], a scheme where Edge datacenters re-direct their requests to other adjacent data centers whenever they become congested. Carlini et al. [7] propose a decentralized sys-

Table 1: Table of Symbols.

Symbol	Meaning
EMC	Edge Mini-cloud
$E = \{EMC_1, \ldots, EMC_m\}$	Set of all the EMCs in the system
$A = \{A_1, \ldots, A_n\}$	Set of all the applications in the system
u_{ij}	Number of users of a_{ij}
U_i	Total number of users of A_i
l_{ij}	Maximum latency experienced by the users of a_{ij}
L_i	Maximum latency admitted by A_i's QoE
$\mathcal{L}(j,k)$	Maximum latency between EMC_j and EMC_k
w_{ij}	Weight (resource occupancy) of a_{ij}
e_{ij}	Energy consumed by a_{ij}
W_j	Max weight that can be sustained by EMC_j
W_j^t	Resource occupancy of EMC_j at time t
$c(a_{ij})$	Function that returns $True$ if a_{ij} is QoE-compliant
$\mathcal{E}_i$	Set of EMCs that can host (w.r.t. QoE) an instance of A_i
$\mathcal{A}_j$	Set of apps that can hosted (w.r.t. QoE) by EMC_j

tem, where autonomous entities in a Cloud Federation communicate to exchange computational services, trying to maximize the profit of the whole Federation. Differently from the previous solutions, in this paper, the efficient exploitation of the resources at the Edge is obtained by optimizing the energy consumption of the system as a whole. As we explain in depth in the rest of the paper, this result is achieved through point-to-point interactions between Edge entities, known as Edge Miniclouds (EMCs). These entities use their communications to detect potential redundant instances of the applications requested by the users. As a result, the users are directed to use only a limited set of instances, thus allowing to shut down the others. However, an user request could be served by a different instance running on another EMC only if the associated QoE constraints remain satisfied. The outcome of the collective behaviour of the entities at the Edge is a notable reduction of the energy needed by the system performing the computational tasks requested by its users.

3 Problem Definition and Proposed Solution

In this paper, we face the problem of how to optimize the execution of applications at the Edge, in order to minimize the energy consumption level of the system as a whole, while respecting the applications' QoE constraints. For the rest of this paper, we will make use of the symbols reported in Table 1.

Specifically, consistently with the definitions of the EU ACCORDION project (https://www.accordion-project.eu/), we consider that the system at the Edge is a federation of so-called *Edge mini-clouds* (EMCs). Each EMC is an entity that supervises a set of other devices with limited resources, like IoT devices, sensors, etc. Applications are sent to an EMC, which is in charge to orchestrate their execution among the devices it controls.

We consider that $E = \{EMC_1, \ldots, EMC_m\}$ is the set of all the EMCs in the system, with $|E| = M$. The set $A = \{A_1, \ldots, A_n\}$ is the set of all the types of applications that can be executed in the system, with $N = |A|$. Each

$A_i \in A$ represents a distinct type of service, with specific requirements in term of resources. In order to meet the requests of the users, several instances of an application A_i can be deployed among the various EMCs. The symbol a_{ij} denotes the instance of the application A_i executed by EMC_j. Running a_{ij} has a weight (in terms of resource occupancy) w_{ij}. This weight is composed of a base weight w_i^{fix} and a variable component w_{ij}^{var}, where the variable component depends on the number of users served by a_{ij}. Therefore, if we denote with u_{ij} the number of users of a_{ij}, we have that $w_{ij}^{var} = u_{ij} w_i^u$, where w_i^u is the weight-per-user of A_i. Thus, $w_{ij} = w_i^{fix} + u_{ij} w_i^u$. The overall number of users served by all the instances of A_i is U_i, while W_j is the maximum weight that can be supported by EMC_j for running all the instances that are assigned to it. In addition to the functional requirements, in order to meet the required QoE, each application has also additional non-functional requirements. These requirements limit the EMCs where an instance can be deployed. We assume that the QoE is expressed in terms of latency, where any service A_i constraints it to be lower than a value L_i. In fact, latency is one of the main factors that influence a user's perception of the quality of a service. Based on this assumption, we also assume that each time a user requests a service, an instance of it is activated on the user's closest EMC (in terms of latency). In this way, the latency is initially minimized. As a consequence, this allocation scheme can also generate a set of redundant instances of the same service. In fact, sets of users initially assigned to different EMCs can be served by just one, properly selected instance, without violating the service's QoE. This allows to shut down the other instances and reduce the amount of energy consumed for serving the same users. Always relaying on the direct intervention of a distant Cloud orchestrator to reach this result could be a source of delay and degradation of the QoE. To overcome this limit, we propose an adaptive self-optimization scheme, based on the autonomous actions of the EMCs. The global goal of the actions of the EMCs' orchestrators is to identify and stop redundant instances of the running applications, thus reducing the system overall energy consumption. The result is achieved by allowing pairs of neighboring EMCs, that share instances of the same application, to evaluate whether they can direct their users to exploit just one of the instances, thus allowing to turn off the other one. In the next, following the pseudocode given in Algorithm 1, we describe the steps executed by a generic EMC_j. We consider that EMC_j has a set $\mathcal{N}$ of neighboring EMCs (EMCs within the communication range of EMC_j). The latency between EMC_j and any of its neighbors $EMC_k \in \mathcal{N}$ is $\mathcal{L}(j,k)$. I_j^t is the set of application types running on the instances on EMC_j at time t. Each application $A_i \in I_j^t$ has a maximum agreed latency L_i, and a set of users u_{ij}, which experiences a maximum latency l_{ij}. At regular time intervals, EMC_j randomly chooses one neighbor $EMC_k \in \mathcal{N}$, using a uniform probability distribution. This distribution is a good baseline for an initial evaluation of the approach, while other choices are left for future works. It then asks EMC_k for the list of its running applications I_k^t with their number of users, its maximum capacity W_k and its actual resource occupancy W_k^t. The solution tries to gather the users of both the instances on the EMC with the lowest actual occupancy.

Algorithm 1 Actions performed by a generic EMC_j at each time step t

Input: $\mathcal{N}$ = set of neighbors of EMC_j

Randomly choose $EMC_k \in \mathcal{N}$
Request I_k^t, W_k, W_k^t to EMC_k
Compute $I_{jk} = \{A_i | A_i \in I_j^t \cap I_k^t\}$
if $I_{jk} \neq \emptyset$ **then**
  if $W_j^t \geq W_k^t$ **then**
    $\mathcal{A}_{jk} = \{A_i \in I_{jk} | \tilde{c}(a_{ik}) = True\}$
    Order $\mathcal{A}_{jk}$ in ascending order using w_{ij}
    Let m be the index of the first application $A_m \in \mathcal{A}_{jk}$ s.t. $W_k^t + w_m^u u_{mj} \leq W_k$
    Direct the users of a_{mj} to use a_{mk}
    Turn off a_{mj}
  else
    $\mathcal{A}_{jk} = \{A_i \in I_{jk} | \tilde{c}(a_{ij}) = True\}$
    Order $\mathcal{A}_{jk}$ in ascending order using w_{ik}
    Let m be the index of the first application $A_m \in \mathcal{A}_{jk}$ s.t. $W_j^t + w_m^u u_{mk} \leq W_j$
    Direct the users of a_{mk} to use a_{mj}
    Tell EMC_k to turn off a_{mk}
  end if
end if

$I_{jk} = \{A_i | A_i \in I_j^t \cap I_k^t\}$ is the set of shared applications. If $I_{jk} \neq \emptyset$, s is the *source* EMC (the one from which the users will be moved), with d the EMC receiving that users. Thus, $W_s^t \geq W_d^t$. EMC_j builds a set $\mathcal{A}_{sd} = \{A_i \in I_{sd} | l_{is} + \mathcal{L}(s,d) \leq L_i\}$ containing the instances whose users can be moved without violating the QoE. EMC_j traverses $\mathcal{A}_{sd}$ in descending order (on the basis of the instances weights in EMC_s), chooosing for the exchange the first application whose users can be transferred without exceeding W_d. In the special case where both the EMCs select each other for an exchange, having equal loads and selecting the same service, the EMC with the lowest ID rejects to receive the exchange, asking for another application. Once the users are directed to another instance, the instance on EMC_s is turned off. This action allows both to save energy and to free space for other potential exchanges or new instances.

4 Experimental Evaluation

In this section, we present a validation of the proposed solution. The results are obtained through a simulation of a target scenario. We use PureEdgeSim [17], a discrete-event simulator for Edge environments, that well matches the EMC-based structure of our scenario, allowing also to easily measure energy consumptions. While each EMC is composed of a set of heterogeneous Edge devices, it is viewed as a single entity, resulting from the aggregation of the resources of the devices it manages. At the beginning of the simulation, each user requests a single application to its closest EMC. In case an instance of the requested

Table 2: w^{fix} and w^u for each application type

Type	VCPU	Ram	BW	VCPU(user)	Ram(user)	BW (user)
Balanced	1	200	20	1 each 10 users	20	2
Comp Bound	2	200	20	1 each 5 users	20	2
Mem Bound	1	400	20	1 each 10 users	40	2
I/O Bound	1	200	40	1 each 10 users	20	4

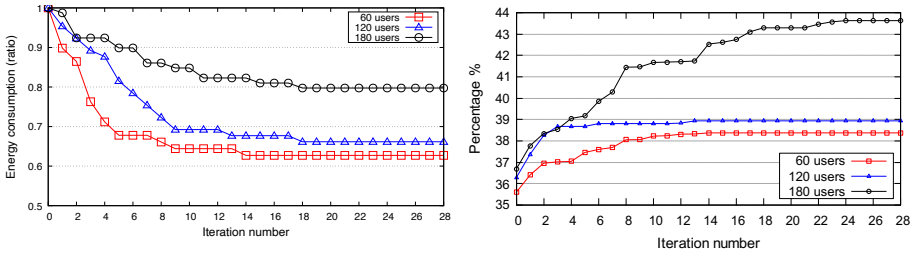
application type already exists on that EMC, the user is simply added to the instance's local set of users.

In the next, we present results coming from different experiments. In each experiment the number of users varies in the set $\{60, 120, 180\}$. Each user device is placed randomly in a bi-dimensional area of 200×200 metres. In all the experiments the number of EMCs is fixed to 4. They are placed at predefined locations inside the simulation space. We assume that any EMC can host any type of application. Moreover, each EMC is able to communicate with the others. There are three types of resources available in the system (at the EMCs): the number of VCPU; the amount of Ram; the amount of network bandwidth (BW).

In the simulations, we use four different types of applications. Application types differ on the resources they request and, as a consequence, the energy footprint they produce when their instances are executed. The application types are divided as *Computational Bound* (i.e, computational intensive), *Memory Bound* (memory intensive) and *I/O Bound* (networking intensive) applications, where "intensive" means having double the requirements of the basic *Balanced* application type. The load of an EMC is calculated as the mean of the percentage of availability of the three resources. The fixed weight w_i^{fix} and the weights per user w^u associated with the different application types is shown in Table 2. Ram and BW are in Mbytes and Mbit/s, respectively. In addition to these parameters, we also use three different values for the maximum application latency L_i: 0.2, 0.3, 0.5 seconds. Therefore, we have 12 possible combinations of parameters for the applications: 4 types of applications times 3 different latency constraints. All the results presented in the next are the average of 10 independent runs.

The first and main result of our evaluation is presented in Figure 1a. This figure presents the evolution over time of the energy required by all the EMCs in the system, including the energy needed for inter-EMCs communications. The results are presented as the ratio between the energy needed at a time $t > 0$ and the energy consumed by the system at the beginning of the simulation. It is possible to observe that the level that is required to serve the very same number of users drops by a minimum of 20% (with 180 users) up to nearly 40% (with 60 users); this drastic reduction in the energy footprint of the system demonstrates the high level of efficiency of the proposed approach.

In Figure 1b we investigate how the configurations adopted by the system are able to remain compliant with the applications' QoE. A simulated latency function is calculated for each user's device, which is the composition of a fixed



(a) (b)

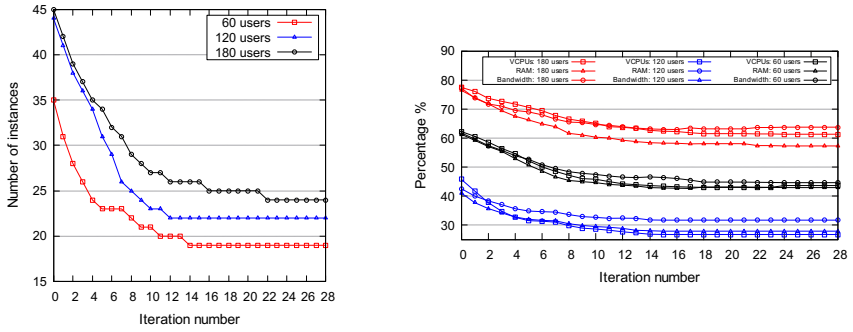
Fig. 1: Temporal evolution of the levels of energy (a) and average latency (b)

part, which is dependent from the communication channel type, and a linear part, proportional to the Euclidean distance between the EMC hosting the instance of the serving application and the user's device. The average latency is measured as the percentage of the maximum average latency, as constrained by the applications' limits. It is possible to note that there is only a limited increase on the average latency. Therefore, the proposed solution shows its ability to remarkably reduce the energy needed to run the instances that serve a given population of users, while remaining well below the limits of the required QoE.

In order to better understand how these results are achieved, the next set of figures analyses how the system collectively adapts its behaviour and how it changes the exploitation of the available resources. Specifically, Figure 2a presents the variation over time of the number of running instances in the system. Clearly, these entities are the source of energy consumption. The ability of the system to detect and eliminate redundant instances is the basis for the energy minimization scheme. It is possible to observe a clear and sharp decrease of this quantity. The final number is nearly the half of the original number of instances. Figure 2b presents the global level of exploitation of the resources. The y axis presents the percentage of all the resources that are required to run the application instances. As in the previous case, we can observe a clear reduction. The amount of this reduction is lower than that of the number of instances, since users are moved from a redundant instance to an active one. As we highlighted in Section 3, each user bring an additional cost in terms of resources. Despite this fact, the overall level of occupied resources is decremented, since the fixed costs needed for running redundant instances are saved.

5 Conclusions

This paper presents a solution for application placement performing edge-to-edge exchanges to reduce the resource usage, while guaranteeing the QoE of applications by keeping the communication latency below given thresholds. The paper provides a definition of the problem and the pseudo code of the proposed approach. An experimental evaluation via simulation shows the validity of our



(a) (b)

Fig. 2: Variation of the total number of instances (a) and system resource loads (b), over time

solution. While the solution is quite a promising one, there is space to improve the results in the near future. It is worth e.g. considering alternative local search criteria and heuristics for the selection criteria of the EMC and application for the swap proposal. This may improve the asymptotic cost savings and is likely to improve the achieved savings as well as the convergence speed of our algorithm.

References

1. Anastasi, G.F., Carlini, E., Dazzi, P.: Smart cloud federation simulations with cloudsim. In: Proceedings of the first ACM workshop on Optimization techniques for resources management in clouds. pp. 9–16 (2013)
2. Aral, A., Ovatman, T.: A decentralized replica placement algorithm for edge computing. IEEE Trans. on Network and Service Management **15**(2), 516–529 (2018)
3. Baraglia, R., Dazzi, P., Guidi, B., Ricci, L.: Godel: Delaunay overlays in p2p networks via gossip. In: IEEE 12th International Conference on Peer-to-Peer Computing (P2P). pp. 1–12. IEEE (2012)
4. Beraldi, R., Mtibaa, A., Alnuweiri, H.: Cooperative load balancing scheme for edge computing resources. In: 2017 Second International Conference on Fog and Mobile Edge Computing (FMEC). pp. 94–100. IEEE (2017)
5. Bruno, R., Conti, M., Mordacchini, M., Passarella, A.: An analytical model for content dissemination in opportunistic networks using cognitive heuristics. In: Proceedings of the 15th ACM international conference on Modeling, analysis and simulation of wireless and mobile systems (2012)
6. Carlini, E., Coppola, M., Dazzi, P., Laforenza, D., Martinelli, S., Ricci, L.: Service and resource discovery supports over p2p overlays. In: 2009 International Conference on Ultra Modern Telecommunications & Workshops. pp. 1–8. IEEE (2009)
7. Carlini, E., Coppola, M., Dazzi, P., Mordacchini, M., Passarella, A.: Self-optimising decentralised service placement in heterogeneous cloud federation. In: 2016 IEEE 10th Int. Conf. on Self-Adap. and Self-Org. Syst. (SASO). pp. 110–119 (2016)
8. Carlini, E., Ricci, L., Coppola, M.: Integrating centralized and peer-to-peer architectures to support interest management in massively multiplayer on-line games. Concurr Comput **27**(13), 3362–3382 (2015)

9. Dazzi, P., Mordacchini, M.: Scalable decentralized indexing and querying of multi-streams in the fog. Journal of Grid Computing **18**(3), 395–418 (2020)
10. Ferrucci, L., Ricci, L., Albano, M., Baraglia, R., Mordacchini, M.: Multidimensional range queries on hierarchical voronoi overlays. Journal of Computer and System Sciences (2016)
11. Gennaro, C., Mordacchini, M., Orlando, S., Rabitti, F.: Mroute: A peer-to-peer routing index for similarity search in metric spaces. In: 5th VLDB International Workshop on Databases, Information Systems and Peer-to-Peer Computing (DBISP2P 2007) (2007)
12. Kavalionak, H., Gennaro, C., Amato, G., Vairo, C., Perciante, C., Meghini, C., Falchi, F.: Distributed video surveillance using smart cameras. Journal of Grid Computing **17**(1), 59–77 (2019)
13. Li, C., Wang, Y., Tang, H., Zhang, Y., Xin, Y., Luo, Y.: Flexible replica placement for enhancing the availability in edge computing environment. Computer Communications **146**, 1–14 (2019)
14. Lulli, A., Carlini, E., Dazzi, P., Lucchese, C., Ricci, L.: Fast connected components computation in large graphs by vertex pruning. IEEE Transactions on Parallel and Distributed systems **28**(3), 760–773 (2016)
15. Maia, A.M., Ghamri-Doudane, Y., Vieira, D., de Castro, M.F.: Optimized placement of scalable iot services in edge computing. In: 2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM). pp. 189–197 (2019)
16. Marzolla, M., Mordacchini, M., Orlando, S.: A p2p resource discovery system based on a forest of trees. In: 17th International Workshop on Database and Expert Systems Applications (DEXA'06). pp. 261–265. IEEE (2006)
17. Mechalikh, C., Takta, H., Moussa, F.: Pureedgesim: A simulation toolkit for performance evaluation of cloud, fog, and pure edge computing environments. In: 2019 Int. Conf. on High Perform. Comput. Simul. (HPCS). pp. 700–707 (2019)
18. Mordacchini, M., Conti, M., Passarella, A., Bruno, R.: Human-centric data dissemination in the iop: Large-scale modeling and evaluation. ACM Trans. Auton. Adapt. Syst. **14**(3) (Feb 2020)
19. Mordacchini, M., Dazzi, P., G.Tolomei, Baraglia, R., Silvestri, F., Orlando, S.: Challenges in designing an interest-based distributed aggregation of users in p2p systems. In: 2009 IEEE ICUMT. pp. 1–8. IEEE (2009)
20. Mordacchini, M., Passarella, A., Conti, M., Allen, S.M., Chorley, M.J., Colombo, G.B., Tanasescu, V., Whitaker, R.M.: Crowdsourcing through cognitive opportunistic networks. ACM Trans. Auton. Adapt. Syst. **10**(2) (Jun 2015)
21. Ning, Z., Dong, P., Wang, X., Wang, S., Hu, X., Guo, S., Qiu, T., Hu, B., Kwok, R.: Distributed and dynamic service placement in pervasive edge computing networks. IEEE Transactions on Parallel and Distributed Systems (2020)
22. Ricci, L., Genovali, L., Carlini, E., Coppola, M.: Aoi-cast in distributed virtual environments: an approach based on delay tolerant reverse compass routing. Concurrency and Computation: Practice and Experience **27**(9), 2329–2350 (2015)
23. Salaht, F., Desprez, F., Lebre, A.: An overview of service placement problem in fog and edge computing. ACM Comput. Surv. **53**(3) (2020)
24. Santoso, G.Z., Jung, Y.W., Seok, S.W., Carlini, E., Dazzi, P., Altmann, J., Violos, J., Marshall, J.: Dynamic resource selection in cloud service broker. In: 2017 Int. Conf. on High Perform. Comput. Simul. (HPCS). pp. 233–235. IEEE (2017)
25. Taleb, T., Samdanis, Kand Mada, B., Flinck, H., Dutta, S., Sabella, D.: On multi-access edge computing: A survey of the emerging 5g network edge cloud architecture and orchestration. IEEE Communications Surveys Tutorials **19**(3), 1657–1681 (2017)