Enriching Remote Control Applications with Fog Computing

Claudio Fiandrino, Paolo Giaccone, Ahsan Mahmood, and Luca Maioli

Abstract Fog computing has emerged in the recent years as a paradigm tailored to serve geo-distributed applications requiring low latency. Remote Control (RC) applications allow a mobile device to control another device from remote. To enrich Quality of Experience (QoE) of RC applications, in this paper we investigate the use of fog computing as a viable platform to offload computation of tasks that would be expensive if performed locally on a mobile device. The proposed approach, supported with next 5G communication systems, will enable a Tactile Internet experience. In this paper we study and compare offload policies to accommodate tasks in the fog platform and analyze the requirements to minimize outages.

Key words: Fog computing; Mobile edge computing; Cellular networks.

1 Introduction

Mobile cloud applications are nowadays essential in our day lives. Among the others, they are used for business and entertainment purposes. However, mobile devices such as smartphones, laptops and wearables are resource constrained, i.e., they have limited energy and computing capabilities at disposal. Mobile cloud and fog computing paradigms overcome such issue through offloading [4, 11, 19]. Offloading is a technique applied to traffic [8] or computation [14]. Computation offloading augments the capabilities of mobile devices by moving the processing of tasks to the cloud. This allows the mobile devices to i) prolong the battery lifetime as process-

Claudio Fiandrino
Imdea Networks Institute, Madrid, Spain, e-mail: `claudio.fiandrino@imdea.org`.
Claudio developed this work as a PhD student at the University of Luxembourg.

Paolo Giaccone · Ahsan Mahmood · Luca Maioli
Dip. di Elettronica e Telecomunicazioni, Politecnico di Torino, Italy e-mail: `firstname.lastname@polito.it`

ing heavy tasks locally is energy-costly, ii) to run sophisticated tasks that the local limited processing capabilities would not permit [12].

Remote Control (RC) applications allow a mobile device, typically a smartphone, to control remotely another device such as a drone or a robot [15]. Typically the control requires line of sight, however with future 5G networks the requirement will not be necessary anymore. RC applications allow the controller to perform actions by means of the controlled device, e.g., taking a picture from a drone. Being resource constrained, batteries and computing capabilities of both controller and controlled devices are limited, thus enriching the applications with functionalities that are non strictly necessary is difficult. For example, performing object detection over a picture taken by the drone can be prohibitive for both the drone and the smartphone. Similarly, performing on-demand statistical analysis may improve robot and automation systems, but it can only be feasible with the help of the cloud [10]. Through offloading, the task becomes then feasible. However it is important that communication overhead is compatible with the small latency required by real-time RC applications. For example, RC applications like drone control require latency to be in the order of ms [3].

The expected requirements for future fifth-generation (5G) wireless systems foresee an improvement in latency from 15 ms of current 4G networks to 1 ms [2]. Levering the ultra-responsive connectivity of 5G systems, RC applications will become an important part of the Tactile Internet vision [18]. Tactile Internet aims at shifting the current content-delivery paradigm of the Internet into a skill-delivery. The ultimate goal is to create a medium capable of transporting touch and other senses. To illustrate with an example, real-time robot applications may suffer service inefficiency (e.g., environment recognition), which can be compensated with human expertise (e.g., through video streaming on RC applications).

In this paper we advocate the use of fog computing to enrich the Quality of Experience (QoE) of RC applications in view of enabling a ultra-responsive Tactile Internet experience. In more details, we investigate the process of allocating computing resources required to perform the tasks[1] such as object recognition that augment the QoE. This process is network-aware, i.e., to maintain a tight synchronization between the controller and the controlled device, communication resources need to be carefully selected. In this work we rely on existing technologies, and not on 5G systems, to verify feasibility constraints. We categorize the tasks according to their suitability to be offloaded in the fog platform or not. Specifically, we identify tasks that must be offloaded in the fog platform because of time constraints, tasks that can be executed either in the fog or in the remote cloud, and tasks that cannot be offloaded in the fog platform because of privacy concerns or because they require external and proprietary software to be executed.

[1] In the reminder of the paper, we use the terms tasks and jobs interchangeably.

2 A Primer on Fog Computing

The concept of fog computing was initially proposed by Cisco to render the network edge capable of cloud computing [5], hence it is also known as *edge computing*. Fog computing is specifically designed to cater for geographically distributed applications which have stringent low latency requirements and context awareness [17]. Consequently it is foreseen that fog computing will play a major role in the development of Internet of Things (IoT) [6]. To this end, various research efforts are being conducted on fog computing platforms supporting IoT applications to assess their efficacy and resource management [1, 16]. The IoT front-end consists of mobile devices which have different computing, networking and storage capabilities. Local processing units called *cloudlets* can be utilized for temporary storage and processing [7]; these include desktop PCs or notebooks. The cloud, which centralizes the processing and backup, can receive the aggregated data from the cloudlets. In vehicular networks, [13] proposes a fog computing approach to place content caches at the network edges; the approach is shown to perform remarkably well as compared to the centralized caching approach, in particular for location-specific applications.

3 Resource Allocation in the Fog Platform

The aim of this section is to present the problem of resource allocation for task offloading in the fog platform. First, the general architecture and the main interacting entities are described. Afterwards, the classification of the jobs that may must be offloaded on the remote platforms is introduced. Lastly, the offload policies for assigning the jobs either to the edge platform or to the external cloud are presented.

Network Architecture

In RC applications, the end-user devices, being resource constrained, are unable to enhance the QoE perceived by the users. One viable solution is to offload the computationally intensive tasks to external resources. Therefore, we envision a network topology where computational platforms are placed both in the cloud and the edge.

The general network architecture, as shown in Fig. 1, is divided into three levels. At the first (user) level, we have a generic user equipment, typically a tablet or a smartphone, acting as controller, and a remote controlled device, typically a drone or a robot. We assume both the devices are connected to the second (access) level of the network through three possible access networks: (i) a generic WiFi network, e.g. through an access point connected to the ADSL access network of an ISP; (ii) an LTE cellular network; (iii) a telecommunication operator's public WiFi network. The last kind of network has been recently introduced by the operators in addition to their existing cellular infrastructure [4]. We assume that edge servers, which users can access to offload some of their jobs, are only available inside the LTE and the operator's WiFi networks.

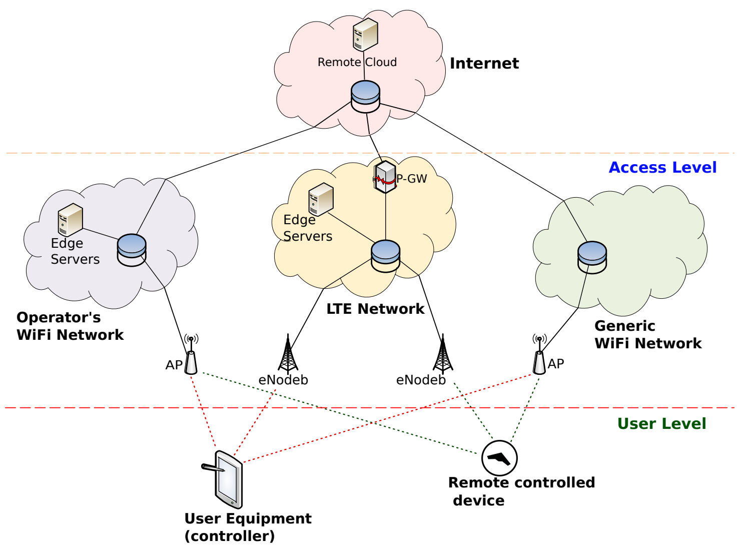


Fig. 1 Network architecture and topology enabling fog-computing capabilities in LTE and operator's WiFi networks.

Finally, the third level in the considered network architecture consists of the global Internet where external cloud servers are available for the execution of users' tasks as well.

Job Definition

In the following, we concentrate just on the jobs that must be offloaded, assuming that other tasks are running locally in the user devices. Each job is associated with a maximum allowed delay for its processing, denoted as *job deadline*. We assume that the the users offload the jobs to the edge servers present in the LTE network or in the public WiFi network, or to the external cloud present in Internet. The range of jobs requested to be processed remotely depends on the scenario as well as on the required processing capabilities. Example of such remote jobs are the following: processing of raw data acquired by a robot, elaboration of digital media (pictures or videos) taken by a drone, object recognition, tasks that require external softwares and services of third-party companies. Moreover, there can also be latency constraints associated with some jobs, limiting them to be processed only at the edge instead of the cloud.

Taking into account the wide range of applications and the latency constraints, we define three classes of jobs as explained in Table 1. Class C jobs are typically the ones requiring third-party software or services, hence such jobs can only be executed in the cloud. Class E jobs can be executed only in the edge platform due to strict

Table 1 Job classification

JOBS CLASS	DESCRIPTION
C (Cloud)	Job can be executed only in the cloud
E (Edge)	Job can be executed only in the edge servers
H (Hybrid)	Job can run either in the cloud or in the edge servers

latency constraints. Finally, class H jobs are the ones with no latency constraints, so they can be run either in the cloud or in the edge platform.

When a new job arrives, there may not be enough resources available to run the job and thus the offload allocation fails. This is denoted as *blocking event*.

Offload Allocation Policies

The allocation policy is responsible to choose where to run a particular job that must be offloaded, based on two possible options: either on the edge servers, or on the cloud. Based on our previous job classification, we consider only class H jobs, for which the allocation decision is not immediate. In our work we formulate the following four policies in order to handle class H jobs:

- *First Fit* (FF): Class H jobs are assigned to the edge servers as long as the resources are available, otherwise they are assigned to the external cloud. The idea is to exploit the resources in the edge servers as much as possible in order minimize delays. As a consequence, the edge servers may become saturated, due to the limited resources, and cause blocking of class E jobs.
- *All Away* (AA): Class H jobs are allocated to the external cloud. The aim is to keep the resources, in the edge servers, available for class E jobs.
- *Load Based* (LB): Class H jobs are allocated to the external cloud only if the load on the edge servers is over 50 percent, otherwise they are executed in the edge servers.
- *Balanced Split* (BS): If the resources are available at the edge, class H jobs are allocated to the edge servers with 0.5 probability, otherwise they are allocated to the external cloud.

4 Performance Evaluation

We developed an ad-hoc event-driven simulator in C language in order to compare the different offload allocation policies for a generic job arrival process.

The simulator was developed to capture the unique features of the proposed system model explained in Section 3. Based on the network architecture shown in Fig. 1, we deduced a simplified network topology, in which we associated a constant communication delay between any pair of entities involved in the communication process.

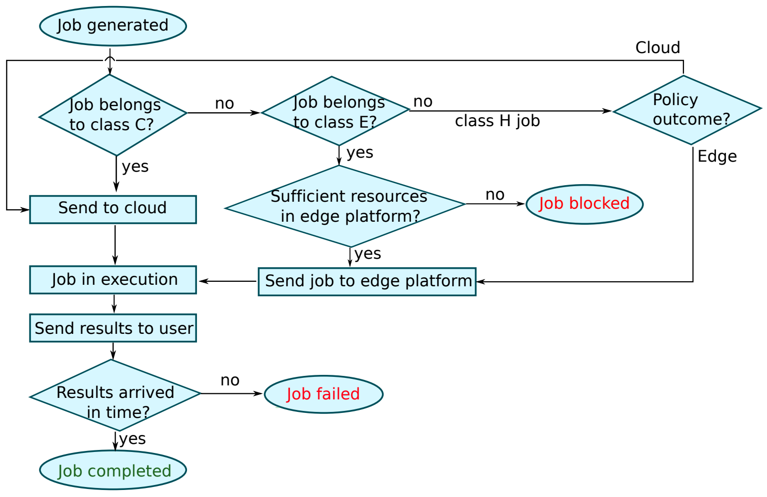


Fig. 2 Workflow of the simulator

The workflow of the simulator is shown in Fig. 2. The first step is the random generation of a job that belongs to one of the classes defined in Table 1. In the simulations, we assume that all classes of jobs arrive with an equal probability, following a Poisson process. We also assume that the cloud has always enough resources to accommodate any number of jobs.

If the generated job belongs to class C, it is sent to the external cloud for execution. After the processing of the job, the results are delivered to the user. If the computation results reach the user within the job deadline, the job is *completed*, otherwise it is considered *failed*. If the generated job belongs to class E, the simulator checks the available resources in the edge servers. If not enough resources are available, the job is *blocked*. Otherwise, the job is sent to the edge servers and the results are sent back to the user, and checked if they satisfy the deadline. Lastly, in the specific case of class H job, the offload policy, chosen among the four ones described in Section 3, determines whether the job should be executed in the edge platform or the cloud.

Table 2 describes all the set of the parameters used for our evaluation. The parameters are divided into three groups: network, edge platform and job related parameters. As shown in Fig. 1, the user devices are connected with the operator's WiFi, LTE and the generic WiFi networks present in the access level. Each of the three networks presents a different access scenario characterized by different parameters. Therefore, we define the network parameters, such as delay, uplink rate and downlink rate, separately for each access network.

As mentioned, the jobs can be allocated either to the edge platform or the cloud. Differently from the cloud, equipped with unlimited computational resources, the

Table 2 Simulation Parameters

NETWORK PARAMETERS		
Parameter	**Description**	**Value**
LTE delay	LTE access and core network delay	5–9 ms
OP-WiFi delay	Operator's WiFi access and network delay	10–20 ms
G-WiFi delay	Generic WiFi access and network delay	8–20 ms
Internet delay	Delay in a one way trip over the Internet	15 ms [a]
Tx G-WiFi	Uplink rate of the generic WiFi network	7.2 Mbps
Rx G-WiFi	Downlink rate of the generic WiFi network	20 Mbps
Tx LTE	Uplink rate of LTE network	3.3–5 Mbps [b]
Rx LTE	Downlink rate of LTE network	20–50 Mbps [b]
Tx OP-WiFi	Uplink rate of the operator's WiFi network	3.2–4.4 Mbps
Rx OP-WiFi	Downlink rate of the operator's WiFi network	8.8–9 Mbps
EDGE PLATFORM PARAMETERS		
Parameter	**Description**	**Value**
Max servers	Number of edge servers	20
Max VCPUs	Number of VCPUs available on each edge server	10
VCPU speed	Processing speed of a single VCPU	1400 MIPS [c]
JOB PARAMETERS		
Parameter	**Description**	**Value**
Job load	Processing requirements of a job	700 MI (million instructions)
Input size	Amount of data uploaded while requesting a job	3 MB
Output Size	Amount of data downloaded while receiving results	1.1 MB
Deadline	Maximum amount of time at disposal to process and deliver the job successfully	7 sec

[a] Verizon IP latency statistics. Available at: `http://www.verizonenterprise.com/about/network/latency/`

[b] Verizon offered speed on LTE. Available at: `http://www.verizonwireless.com/mobile-living/network-and-plans/4g-lte-speeds-compared-to-home-network/`

[c] Amazon m3.medium vcpu benchmark. Available at: `https://s3.amazonaws.com/cloudharmony/geekbench3_3_1_6/aws:ec2/m3.medium/ebs/sa-east-1/2014-11-12/2636/369105-9/geekbench.html`

edge platform is composed of a limited number of servers available in the LTE network and in the operator's WiFi network. For each edge server, we define the number of virtual CPUs (VCPUs) and their processing capabilities.

Finally, we have the job related parameters which depend on the kind of task required to be processed. We assume a generic image processing job, where a user sends a raw image to the edge platform or the cloud and requests to send the processed image to the other user. The selected parameters in Table 2 reflect such choice.

Given an allocation policy, 5000 simulation runs are executed. In each run, the job arrival rate is increased, ranging from 1 to 5000 jobs per second. In addition, for each of the simulation parameters that takes a range of values in Table 2, we generate a uniformly distributed value in the considered range.

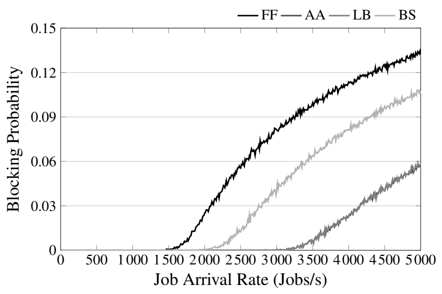


Fig. 3a Global blocking probability

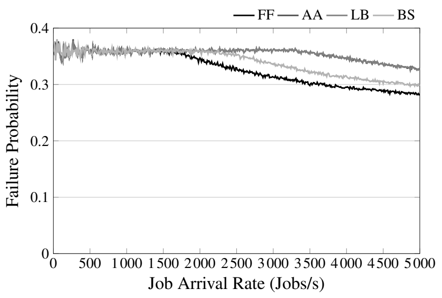


Fig. 3b Global failure probability

4.1 Numerical Results

We start by comparing the four offload policies in order to find out the best policy among them. Then, we perform the server utilization analysis to evaluate the minimum number of servers required in the edge platform for a given job arrival rate.

Offload Allocation Policy Comparison

The allocation policies are compared in terms of their global blocking probability and global failure probability as shown in Figs. 3a and 3b respectively. The global blocking probability represents the fraction of *blocked jobs*, while the global failure probability represents the fraction of *failed jobs*, in the overall system. As a reminder, the First Fit (FF) policy allocates the class H jobs to the edge platform with a higher priority. This causes the edge servers to saturate sooner (in comparison to the other policies) and thus increasing the number of blocked jobs. However, since more jobs get executed at the edge, the number of failed jobs reduces. On the other hand, the All Away (AA) policy allocates all of the class H jobs to the cloud. As a consequence, the number of blocked jobs are reduced, but at the expense of the increase in the number of failed jobs. Therefore, AA and FF are the best policies in terms of global blocking probability and global failure probability respectively. The LB policy, which assigns jobs to the edge servers based on a given load, behaves similarly to AA for high load. Indeed, when the job arrival rate is high, the load of the class E jobs on the edge servers is high as well. Thus, almost all of the class H jobs are allocated to the cloud. The behavior of the BS policy remains in between FF and AA.

In order to find the overall best policy, we have computed the summation of global blocking and failure probabilities (i.e. 1 minus the probability that the job is completed) and based on this we have observed that AA policy is the best among all the other offload policies. Thus, for the following investigations we will consider only AA policy.

Fig. 4 shows the blocking and the failure probabilities in LTE and WiFi networks. It is evident that the failure probability of WiFi networks is higher than that of LTE network, due to the fact that the overall delay of WiFi network is greater than the

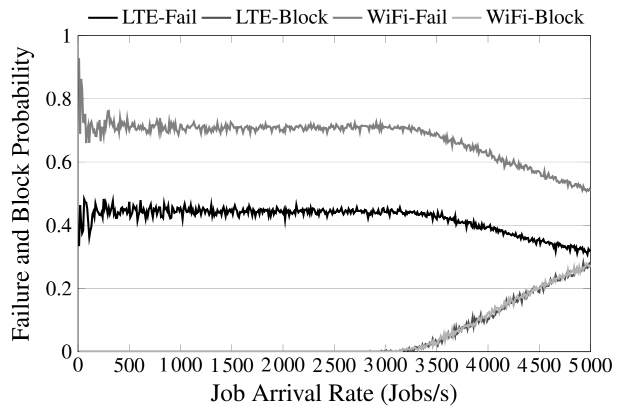


Fig. 4 Failure and blocking probabilities for LTE and WiFi networks for the All Away (AA) offloading policy

delay of LTE network. As a result, the processed jobs traversing WiFi networks are more likely to arrive after the deadline. In addition, the downlink rate of LTE is also higher as compared to that of the WiFi networks. Moreover, for low job arrival rate, the blocking probability is zero in both LTE and the WiFi networks. This is because there are sufficient resources available in the edge servers to accommodate class E jobs, while both class C and class H jobs are allocated to the cloud. Once the resources are exhausted, at a job arrival rate of about 3200 jobs/sec, class E jobs begin to get blocked and the blocking probability starts to increase. This phenomenon decreases the failure probability as the blocked jobs do not consume additional network and computing resources, thus allowing more jobs that are executed in the cloud to arrive at the user device within the deadline.

Server Utilization Analysis

This second part of results focuses on the server utilization in the edge platform. Our design aim is to quantify the number of servers required to sustain a given job arrival rate while maintaining the blocking probability under 1%. Here, we are only interested in the blocking probability as it depends on the capacity of the edge platform. The failure probability, on the other hand, depends mainly on the network performance. Our previous results have shown that AA policy outperforms the other policies in terms of blocking probability. However, to evaluate the worst case scenario, useful for a safe design of the edge platform, we consider the FF policy which gives the highest blocking probability, according to Fig. 3a.

We assume that the pseudo-sinusoidal traffic pattern during 24 hours of a working day, as measured in [9], is followed by the job arrival rate, as shown in Fig. 5. In the figure, the job arrival rate is normalized to its peak value.

We evaluate the required number of edge servers in two different scenarios: a small community with few users and a peak arrival rate set at 1100 jobs/s; and a large community, similar to a city, with a large number of users and a peak arrival

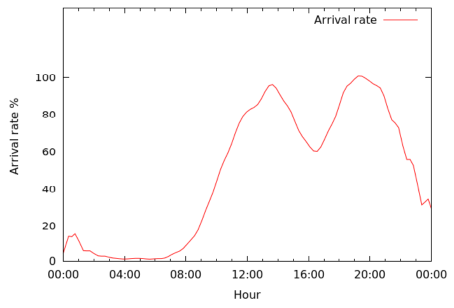
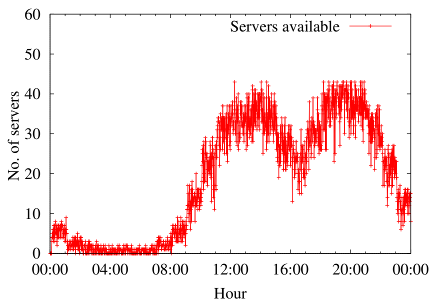


Fig. 5 Normalized arrival rate during 24 hours of a working day



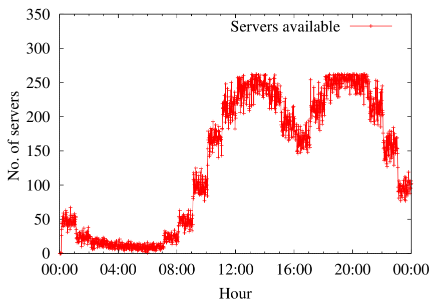


Fig. 6a Small community evaluation, blocking probability=1%, FF policy

Fig. 6b Large community evaluation, blocking probability=1%, FF policy

rate set at 100,000 jobs/s. Fig. 6a shows the number of servers needed to satisfy the job requests at the blocking probability of 1% throughout the day for the small community scenario. In this case, the number of servers required at the peak hour is 43. Similarly, Fig. 6b shows the number of servers for the large community scenario. Here, the number of servers required at the peak hour is 262. It is worth to note the step behavior of the number of required servers, due to the policy that tends to consolidate the usage of the edge servers. Only when all the edge servers are saturated, then a new set of servers is activated to accommodate the new requests.

5 Conclusion

Enriching remote control applications with advanced capabilities can be highly expensive, in terms of required computational power and energy, for a resource constrained user device. Fog computing is a promising alternative to offload computationally expensive jobs away from the user device to the edge. In this paper, we present a network architecture where a communication infrastructure, based on LTE and WiFi networks, is equipped with fog and cloud computing platforms. We categorize the jobs to be offloaded specifically in the edge (E), cloud (C) or in both (H) according to their suitability. Moreover, we define a set of 4 offload policies to manage class H jobs. The All Away (AA) policy outperforms the other polices in terms of the global blocking probability as it offloads all of the class H jobs to the cloud, thus decreasing the load on the edge servers. In terms of the global failure probability, the First Fit (FF) policy performs better than the other policies as it offloads the class H jobs at a higher priority to the edge servers, thus reducing the latency and meeting the deadline. Furthermore, we quantify the number of edge servers required to sustain a given job arrival rate while maintaining the blocking probability under 1%, in two scenarios, consisting of small and large communities.

References

1. Aazam, M., Huh, E.N.: Fog computing micro datacenter based dynamic resource estimation and pricing model for IoT. In: IEEE International Conference on Advanced Information Networking and Applications (AINA), pp. 687–694 (2015)
2. Andrews, J.G., Buzzi, S., Choi, W., Hanly, S.V., Lozano, A., Soong, A.C.K., Zhang, J.C.: What will 5G be? IEEE Journal on Selected Areas in Communications **32**(6), 1065–1082 (2014)
3. Asadpour, M., den Bergh, B.V., Giustiniano, D., Hummel, K.A., Pollin, S., Plattner, B.: Micro aerial vehicle networks: an experimental analysis of challenges and opportunities. IEEE Communications Magazine **52**(7), 141–149 (2014)
4. Balasubramanian, A., Mahajan, R., Venkataramani, A.: Augmenting mobile 3G using WiFi. In: 8th International Conference on Mobile systems, applications, and services, MobiSys, pp. 209–222. ACM (2010)
5. Bonomi, F., Milito, R., Natarajan, P., Zhu, J.: Fog computing: A platform for Internet of Things and analytics. In: Big Data and Internet of Things: A Roadmap for Smart Environments, pp. 169–186. Springer (2014)
6. Bonomi, F., Milito, R., Zhu, J., Addepalli, S.: Fog computing and its role in the Internet of Things. In: 1st Workshop on Mobile Cloud Computing, MCC, pp. 13–16. ACM (2012)
7. Chen, M., Hao, Y., Li, Y., Lai, C.F., Wu, D.: On the computation offloading at ad hoc cloudlet: architecture and service modes. IEEE Communications Magazine **53**(6), 18–24 (2015)
8. Fiandrino, C., Kliazovich, D., Bouvry, P., Zomaya, A.Y.: Network-assisted offloading for mobile cloud applications. In: IEEE International Conference on Communications (ICC), pp. 5833–5838 (2015)
9. Hohwald, H., Frías-Martínez, E., Oliver, N.: User modeling for telecommunication applications: Experiences and practical implications. In: International Conference on User Modeling, Adaptation, and Personalization, pp. 327–338. Springer (2010)
10. Kehoe, B., Patil, S., Abbeel, P., Goldberg, K.: A survey of research on cloud robotics and automation. IEEE Transactions on Automation Science and Engineering **12**(2), 398–409 (2015)
11. Kumar, K., Lu, Y.H.: Cloud computing for mobile users: Can offloading computation save energy? Computer **43**(4), 51–56 (2010)
12. Mach, P., Becvar, Z.: Mobile edge computing: A survey on architecture and computation offloading. IEEE Communications Surveys Tutorials (2017)
13. Malandrino, F., Chiasserini, C., Kirkpatrick, S.: The price of fog: A data-driven study on caching architectures in vehicular networks. In: 1st International Workshop on Internet of Vehicles and Vehicles of Internet, IoV-VoI, pp. 37–42 (2016)

14. Ragona, C., Granelli, F., Fiandrino, C., Kliazovich, D., Bouvry, P.: Energy-efficient computation offloading for wearable devices and smartphones in mobile cloud computing. In: IEEE Global Communications Conference (GLOBECOM), pp. 1–6 (2015)
15. Rouanet, P., Oudeyer, P.Y., Danieau, F., Filliat, D.: The impact of human-robot interfaces on the learning of visual objects. IEEE Transactions on Robotics **29**(2), 525–541 (2013)
16. Sarkar, S., Chatterjee, S., Misra, S.: Assessment of the suitability of fog computing in the context of Internet of Things. IEEE Transactions on Cloud Computing (2015)
17. Sciarrone, A., Fiandrino, C., Bisio, I., Lavagetto, F., Kliazovich, D., Bouvry, P.: Smart probabilistic fingerprinting for indoor localization over fog computing platforms. In: 5th IEEE International Conference on Cloud Networking (CloudNet), pp. 39–44 (2016)
18. Simsek, M., Aijaz, A., Dohler, M., Sachs, J., Fettweis, G.: 5G-enabled tactile internet. pp. 460–473 (2016)
19. Wang, S., Zhang, X., Zhang, Y., Wang, L., Yang, J., Wang, W.: A survey on mobile edge networks: Convergence of computing, caching and communications. IEEE Access (2017)