A heterogeneous mobile cloud computing model for hybrid clouds

Saúl Alonso-Monsalve, Félix García-Carballeira, Alejandro Calderón

PII: S0167-739X(17)31389-4
DOI: https://doi.org/10.1016/j.future.2018.04.005
Reference: FUTURE 4071

To appear in: *Future Generation Computer Systems*

Received date : 28 June 2017
Revised date : 23 February 2018
Accepted date : 1 April 2018

A Heterogeneous Mobile Cloud Computing Model for Hybrid Clouds

Saúl Alonso-Monsalve, Félix García-Carballeira, Alejandro Calderón

Avda. Universidad 30, 28911 Leganés, Madrid, Spain

Computer Science and Engineering Department

University Carlos III of Madrid

Abstract

Mobile cloud computing is a paradigm that delivers applications to mobile devices by using cloud computing. In this way, mobile cloud computing allows for a rich user experience; since client applications run remotely in the cloud infrastructure, applications use fewer resources in the user's mobile devices. In this paper, we present a new mobile cloud computing model, in which platforms of volunteer devices provide part of the resources of the cloud, inspired by both volunteer computing and mobile edge computing paradigms. These platforms may be hierarchical, based on the capabilities of the volunteer devices and the requirements of the services provided by the clouds. We also describe the orchestration between the volunteer platform and the public, private or hybrid clouds. As we show, this new model can be an inexpensive solution to different application scenarios, highlighting its benefits in cost savings, elasticity, scalability, load balancing, and efficiency. Moreover, with the evaluation performed we also show that our proposed model is a feasible solution for cloud services that have a large number of mobile users.

Keywords: fog computing, heterogeneous cloud, hybrid cloud, mobile cloud computing, mobile edge computing, participating device.

1. Introduction

Throughout the last few years, cloud computing (CC) has provided computing solutions to lots of companies, organizations, and individual users in the form of services over the Internet. CC provides on-demand, pay-per-use, and highly scalable computing capabilities for services that enhance the user experience in a transparent way for the user [1]. Meanwhile, with the current exponential growth of mobile devices, there is an emerging concept called mobile cloud computing (MCC) that has erected to integrate CC into the mobile environment [2]. In MCC, user applications are computed in remote clouds rather than in their own mobile devices, providing multiple benefits to the mobile users, such as a longer battery lifetime or a lower processing load.

Among the different approaches to MCC, we can bring the computation capabilities closer to the mobile users. This model locates small-scale servers or cloudlets at the edge of the network (e.g., base stations or coffee shops) in order to avoid latency or bandwidth issues CC experiment. This approach is related to novel paradigms such as fog and mobile edge computing and is supposed to be a key aspect in 5G [3, 4]. On the other hand, it needs a periodic synchronization between the edge servers and the cloud, so several questions arise: when should the edge servers upload data to the cloud servers? How will the cloud handle such amounts of data from multiple edge servers located all over the world? How will these systems guarantee consistency (one of the desired properties of a distributed system according to Brewer's theorem [5])? There are only a few published works related these issues [6, 7] and they are all also theoretical. Besides, this approach has numerous security issues (e.g., authentication, mobility, or access control) [8, 9], and not all companies and organizations will be able to deploy multiple servers at the edge of the network due to the high investment that it entails.

For all these reasons, we have developed a heterogeneous mobile cloud computing model that can provide most of the benefits of the fog and mobile edge computing solutions, but it can also be deployed easily and inexpensively by enterprises into their current cloud systems. More specifically, our work provides the following contributions:

- A heterogeneous mobile cloud computing model, which combines the current mobile cloud architecture with the utilization of volunteer platforms as resource providers.
- A complete description of this model and how it can be deployed in public, private, and hybrid clouds by using the BOINC open-source software: the devices that form the volunteer platforms should run the BOINC client software, and the cloud side should run the BOINC server software.
- A modeling of the new proposed model using ComBoS, an open-source simulator for volunteer computing and desktop grids created by the authors, as an entry point.
- An explanation of the benefits of our solution, including cost savings, elasticity, scalability, load balancing, and efficiency.
- An extensive simulation-based evaluation considering several realistic scenarios that demonstrates that our proposed model is a feasible solution for different cloud services.

The rest of the paper is organized as follows: Section 2 describes the background and discusses related work; Section 3 introduces in detail our proposed MCC model; Section 4 analyzes the performance of our model applied to different services; and finally, Section 5 concludes the paper and presents some future work.

2. Background Related Work

In this section, we describe the background and present the work related to the solution proposed in this paper. In particular, Section 2.1 is about mobile cloud computing, while Section 2.3 deals with volunteer computing.

2.1. Mobile Cloud Computing

Mobile cloud computing (MCC) is a concept that refers to the integration of cloud computing into the mobile environment [2]. In this way, MCC allows for a rich user experience; since client applications run remotely in the cloud infrastructure, applications use fewer resources in the user's mobile devices. The typical architecture of MCC is shown in Figure 1 [2]. In this figure, the mobile user devices (from Mobile device A to Mobile device F) are connected to the mobile networks through base stations: satellites, access points, or base transceiver stations (BTS). The network operators are the providers of wireless communication services, and they allow the mobile devices to access the cloud via the Internet. This left half of Figure 1 is called the network edge, while the right half, where cloud computing is located, is called the network core. Cloud controllers are located within a cloud, and their job is to manage the user requests and answer them by providing the mobile users with the corresponding cloud services. Even though new types of cloud services have emerged in recent years - such as CaaS (container as a service), DBaaS (database as a service) or even GaaS (game as a service) - cloud services are mainly classified as IaaS (infrastructure as a service), PaaS (platform as a service), and SaaS (software as a service) [10]:

- IaaS: it is the lowest layer of cloud computing. It offers any physical or virtual resource to the clients.
- PaaS: it is the middle layer. It provides the user with the ability to develop and manage applications regardless of the infrastructure they use.
- SaaS: it is the highest layer. It allows the user to consume applications through the Internet using a specific client software.

Among the multiple advantages of MCC, it can improve the user experience [2, 11] in terms of: (1) battery lifetime and (2) lower CPU load, since the processing tasks are performed in the cloud instead of in the mobile device; (3) storage capacity, because files can be stored on remote cloud servers, without consuming the storage resources of the mobile device; and (4) reliability, since data is stored in a number of computers within a cloud, thus preventing data loss. There are MCC applications of many kinds [12]: mathematical tools, file search, imaging tools, games, download applications, security, etc. Examples of MCC applications are Google's Gmail for mobile[1] or Amazon Simple Storage Service (Amazon S3)[2].

According to [13], there are two other definitions of MCC. The first one is shown in Figure 2, where some mobile devices act as cloud resource providers forming a peer-to-peer (P2P) network. In this model, the mobile devices in the local vicinity and other stationary devices (if available) would create an ad-hoc

[1]`https://www.google.com/mobile/mail/`
[2]`https://aws.amazon.com/s3/`

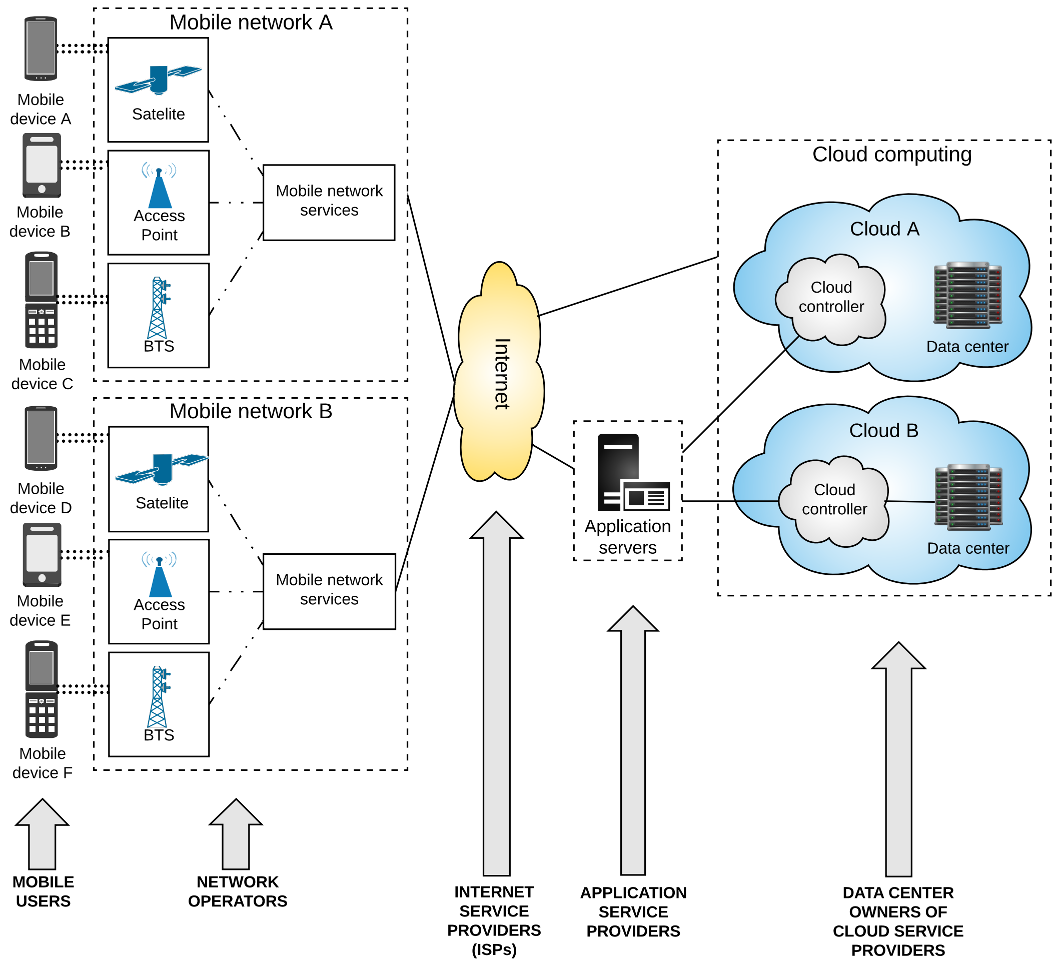


Figure 1: Mobile cloud computing basic architecture, based on [2].

network which can be accessed by other mobile devices in order to run their applications. Theoretically, this model allows offloading the cloud tasks to the mobile devices that form the virtual resource cloud. Besides, latency is also reduced, since the mobile users just have to access the virtual cloud resource instead of traversing lots of hops to get to the remote cloud. Examples of this approach are Hyrax [14] and SATIN [15], but there are no real deployments of such solutions. However, there are different issues related to this model:

- It is not clear how the mobile users will find the mobile devices forming the virtual resource cloud and how these devices are able to process the same tasks as a remote cloud.
- Battery lifetime is a critical issue in mobile devices, so, if the mobile devices of the virtual resource cloud processed complex tasks, their batteries would run out.
- Most clouds need to back up all of the user's information on their servers, so, if mobile devices

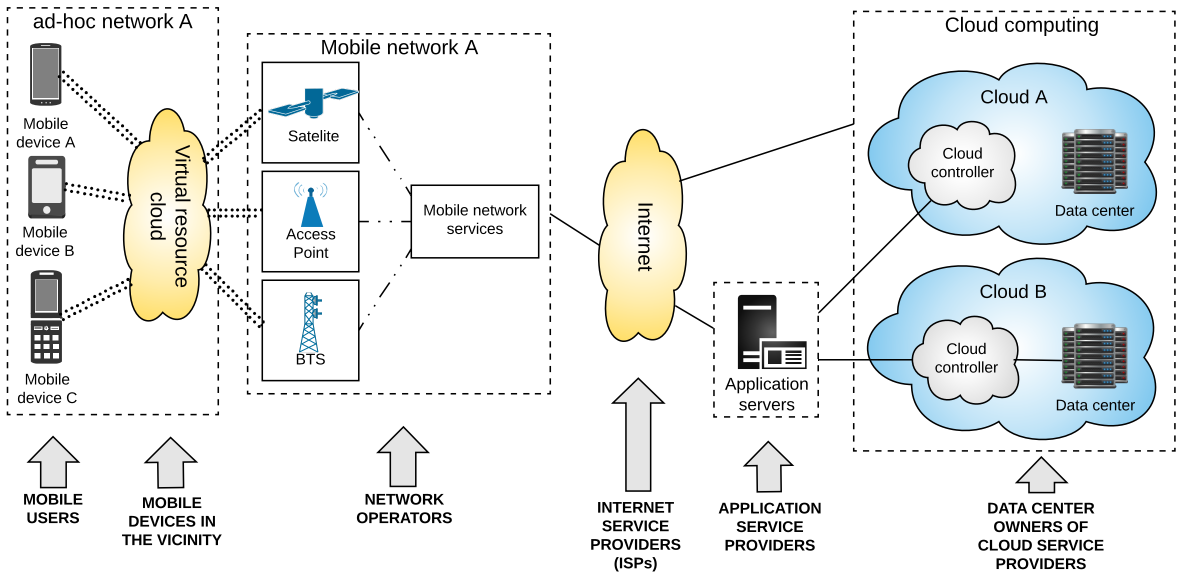


Figure 2: Virtual resource cloud forming a P2P network and acting as a resource provider.

perform the tasks near the mobile users, synchronization with the cloud servers becomes much more complicated.

- The devices that form the virtual resource cloud are untrusted mobile devices, so why would a mobile user send information to them? Nothing can warrant the user that their data will not be treated maliciously.

2.2. Fog and Edge Computing

Together, Fog and Mobile Edge Computing (FMEC) establish a distributed computing paradigm that extends the services provided by the cloud to the edge of the network [16, 17]. In fog computing, a large number of devices of all types access to cloud services. However, much of the processing is done near the edge of the network instead of entirely in the core, taking advantage of the large number of sources on the edge [18]. This FMEC model is gaining relevance in the scientific community, since it will play an important role in 5G [19, 20], and it will allow mobile devices to bypass the latency and bandwidth issues of the current cloud systems, allowing the large number of mobile devices (the company CISCO systems predicted that there will be 50 billion devices with Internet access by 2020 [21], including Internet of Things (IoT) [22] devices) to use the services offered by cloud computing without saturating the cloud servers and networks.

A solution for FMEC is the use of cloudlets [23], which are composed of trusted, resource-rich nodes which are located in the near vicinity of a mobile user. This solution is presented in the last MCC model [13], shown in Figure 3. In this model, cloudlets are used to avoid latency and bandwidth issues related to cloud computing. A cloudlet (also known as edge server or edge cloud) [24, 25] is a small cloud datacenter located at the edge of the network, and it aims to provide resources with low latency to mobile devices. In other words, their goal is to bring the cloud closer to mobile users, by offloading the computations from mobile devices onto virtual machines (VM) [19]. In 2015, researchers from Carnegie Mellon University created OpenStack++ [26], an open-source OpenStack extension that allows the integration of cloudlets in an OpenStack infrastructure, in addition to VM provisioning and handoff. The potential for synergy between the cloudlet concept and Fog [16] and Mobile Edge Computing [27] (FMEC) has been studied in [20, 28].

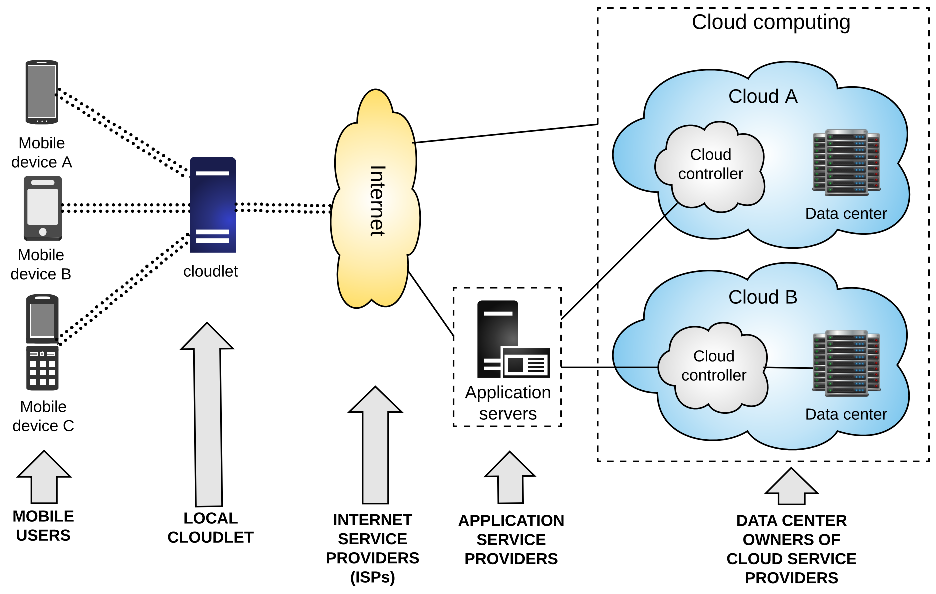


Figure 3: A cloudlet providing public resources to mobile user devices.

In [29] a drop computing paradigm is introduced. This paradigm proposes the concept of decentralized computing over multilayered networks, combining cloud and wireless technologies over a social crowd formed between mobile and edge devices, thus, instead of every data or computation request going directly to the cloud, Drop Computing employees the mobile crowd formed of devices in close proximity for quicker and more efficient access. This solution is similar to the solution proposed in our article. However, the main problem with this solution is how to locate other drop computing nodes to distribute data or computation.

Other work that uses the idea of edge computing is [30], where a middleware between the smartphone and the mobile cloud is described. The idea is that the computational activity used by a smartphone is offloaded to other mobile devices, just as it is now often done to desktop or laptop computers. The device itself then becomes an interface to applications running on different smartphones, and uploads merely raw data and downloads the processed material when it is needed. This solution is also similar to the solution proposed in our article. However, the main problem with this solution is the same as the above work, how to locate the other smartphones to offload the computation.

[31] proposes a code offloading framework, called mCloud, which consists of mobile devices, nearby cloudlets and public cloud services, to improve the performance and availability of the MCC services. This system uses a discovery service responsible for discovering the mobile cloud infrastructure resources. This discovery service is similar to a BOINC server that stores the list of volunteer devices. The advantage of our solution is that we use BOINC as a volunteer computing platform. On the other hand, mCloud depends on a discovery service that detects other available mobile devices in the proximity. The main problem with this solution, as the authors of this paper explain, is the additional overhead an energy consumption of this solution. This problem does not appear in the solution proposed in this paper.

When a large number of devices of all types access to cloud services, resource allocation (compute cycles, temporary memory, etc.) became a crucial issue to adjust cost and to provide the appropriated service. In order to prevent some peak demand, a cloud federation solution [32] could be used. However, in ephemeral events (for example, a holiday week where people move to another place) considerable requirements in a short and temporal period do not justify building a new datacenter, deploying a fast interconnection network, or moving a massive amount of data. Part of the processing could be done near the edge of the network instead of entirely in the core. Extreme edge infrastructures [33] are an interesting proposal. But again the resources near the edge of the network are limited, not always could match the number of client devices requests, and it is not easy to deploy a service quickly and cost-effectively. Anticipation could not always be possible for pre-allocating of any possible workload peak for both: the Edge part of the cloud and the Core part of the cloud. But what if the clients (devices that access to the cloud services) provide their own resources? This on-demand self-allocation strategy could be combined with other solutions as mentioned above, and it is especially recommended (but not limited) for non-persistent services.

The software for managing this proposed on-demand non-persistent elements is a significant challenge. Resource managers for cloud such as Mesos [34] or Kubernetes [35] are used to join semi-persistent devices (existing devices where the number of failures in this set of devices is limited) with a push-like control technique. Volunteer computing was designed with a pull-like control technique in mind in order to work with a volatile (non-persistent), heterogeneous and unreliable resources (where the devices come and go). Studies such as [36] demonstrate that under higher load, a pull-like technique is better for improving scalability.

As we discuss later, volunteer computing is primarily used for scientific projects for free [37], while cloud computing is used for an on-demand paid service workload [38]. However, while Mesos and Kubernetes take care of resource in a transparent way for users, in volunteer computing the users have to configure for which (scientific) projects the device is going to participate, in a flat administration domain. Cloud resource managers (Mesos, Kubernetes, etc.) use to scale up and down an existing service (for load balancing) in a homogeneous pre-existent infrastructure, but volunteer computing projects use to scale up as maximum as possible.

Mesos consists of a master daemon that manages agent daemons running on each cluster node, and the Mesos framework runs tasks on these agents. This resource manager is only available for infrastructures where the number of agents is known. With Kubernetes, the idea is very similar. The objective of our solution is to reduce the number of resources that the cloud provider has to install. The aim is to reduce the number of local resources in the cloud and use volunteer devices instead. This feature allows reducing costs on the cloud side, one of the advantages of our solution.

While Docker, a container-based technology, is proposed as a platform for Edge Computing [39], still is experimental [40] for volunteer computing (as a light way to solve execution on heterogeneous resources). As far as the authors know, there is no proposal for an on-demand self-allocation client-based solution that could be added to Edge and Core cloud for non-persistent services (for example, for mobile IoT [41]).

2.3. Volunteer Computing

Volunteer computing is a type of distributed computing in which ordinary people donate processing and storage resources to one or more scientific projects. Luis Sarmenta coined the term volunteer computing (VC) during his Ph.D. research [42]. BOINC [43] is the main middleware system for VC that makes it possible for scientists to design and operate public-resource computing projects. The applications supported by BOINC are diverse and include communication and large storage data-intensive applications. For computer owners to become volunteers, they have to download and run a BOINC client program on their computers. Each volunteer can participate in multiple BOINC projects. If they choose to do so, they have the freedom to specify how they would like their resources to be allocated among the projects. Examples of BOINC projects include Einstein@Home, Enigma@Home, LHC@Home, MilkyWay@Home, SETI@Home, and Universe@Home.

The basic BOINC architecture has a central server that is responsible for dividing applications into thousands of small independent tasks. As the worker nodes request workunits, the central server distributes the tasks among them. If this server initiated the communications, NAT (Network Address Translation) issues might arise from a bidirectional communication. For this reason, when a worker is ready to submit results or needs more work, it initiates the communication. The centralized servers never initiate the connection with worker nodes.

Moreover, VC can be used on mobile devices for executing tasks. In this kind of platforms, the BOINC application only computes when the device is plugged into a power source (AC or USB), and the battery is over 90% of charge, so it will not significantly reduce the battery life or the recharge time. Besides, BOINC transfers data only when the devices are connected to a WiFi network, and the device screen is off. It is true that mobile devices only perform volunteer computing tasks when they are plugged and almost charged. This situation seems to be unusual; however, most of the people charge their phones overnight. Considering an average sleeping time of 8 hours, and around one or two to hit a 100% of battery, every mobile device can contribute between 6 or 7 straight hours during the night to volunteer computing projects. Also, there are current studies that try to exploit this model by using the idle computing resources of smart TV sets as volunteer nodes [44], which shows that this type of computing can become part of the IoT world.

Apart from BOINC, there are other VC systems, such as WeevilScout [45] and Comcute [46]. Both solutions consist of using web browsers from anonymous users to perform master-slave VC tasks. In fact, the solution described in [46] proposes a multi-level volunteer computing architecture, and it is similar to the approach introduced in [47] since both have volunteer users computing parallel executions. The use of VC systems for Big Data processing has been studied in [48]. In this article, the authors describe an architecture of intelligent agents to optimize Big Data processing. In [49], the authors present a VC solution called FreeCycles, which supports MapReduce jobs. FreeCycles improves data distribution (among mappers and reducers) by using the BitTorrent protocol to distribute data and improves intermediate data availability by replicating files throughout volunteers to avoid losing intermediate data. However, these solutions are not based on BOINC, and they have plenty of future challenges.

The use of VC in cloud computing has been explored in [50]. In this paper, the authors introduce Cloud@Home, a combination of the VC and cloud computing paradigms used for scientific purposes. Cloud@Home consists of creating a cloud by the combination of multiple low-power volunteer nodes. However, this approach is entirely different to ours, because our solution can be applied to any existing cloud system that wants to expand their resources, and it is not an alternative to the current cloud systems, unlike Cloud@Home. Besides, we wanted to propose a new MCC model based on BOINC, because it is the most relevant middleware for VC, and there are currently hundreds of thousands of volunteers participating in their projects.

2.3.1. BOINC

In BOINC, servers are responsible for managing projects. The architecture of the server side is shown in Fig. 3-2. The server side of a project consists of two parts:

- A project back end that supplies applications and workunits, and that handles the computational results. It includes: a work generator, which creates workunits and their corresponding input files; a validator that examines sets of results and selects canonical results; an assimilator that handles workunits that are completed; and a file deleter, which deletes input and output files that are no longer needed.
- A BOINC server complex that manages data distribution and collection. It includes: one or more scheduling servers (sometimes called task servers), that communicate with participant hosts; and data servers, that distribute input files and collect output files. For small projects, if there are no data servers, scheduling servers also operate as data servers.

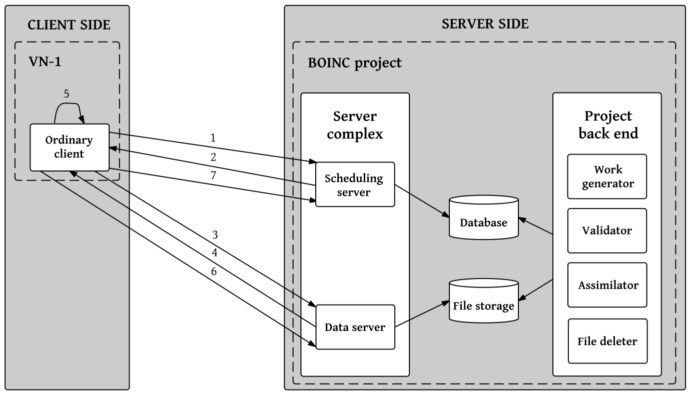


Figure 4: BOINC Architecture.

Figure 4 shows how BOINC currently works. It contemplates a scenario with just one Volunteer Node (VN-1: Ordinary client) and one project. When a volunteer (in this case, VN-1) wants to execute a task, it first makes a request to the Scheduling Server of the project (step 1 of Fig. 3-1). Then, the server answers with a workunit (step 2), which stores the computation to be performed and a list of addresses where the input files needed to perform the computation can be downloaded. Therefore, the next step for the client is to download the necessary files from the list addresses, which correspond to the data servers of the project (steps 3 and 4). Once the client has downloaded the files, it computes the task (step 5). When the execution is finished, the client simply has to send the output files generated during the execution to the data servers (step 6), and report the results obtained to the scheduling server (step 7).

3. Proposed Model

In this section, we describe our solution in detail. More specifically, Section 3.1 defines some basic terms. Section 3.2 outlines the aims and goals of this approach, Section 3.3 shows the architecture of the proposed model, Section 3.4 describes the volunteer platforms that we consider in our solution, Section 3.5 depicts the two main application scenarios, Section 3.7 presents the incentive scheme we propose for the volunteer users, and finally Section 3.8 depicts the security aspects needed.

3.1. Terms

We have split the actors of our model into three types:

- Mobile users: the final clients that consume the cloud services.
- Participating devices: desktop computers or mobile devices that collaborate in a cloud system by donating their idle resources. In other words, they act as intermediate service providers. They form the volunteer platforms.

- Cloud infrastructure: hardware and software components that provide the cloud services, without considering the participating devices.

3.2. Aims and Goals

As we showed in previous work [51], some clouds are experiencing a saturation of their networks and servers due to the high number of user devices accessing the services offered. In fact, this issue is only going to worsen in the next few years because, as we mentioned in Section 2, the company CISCO systems predicted in 2011 that there will be 50 billion devices with Internet access by 2020 [21], and a huge percentage of these devices is going to access mobile cloud services. Some solutions from previous literature provide mechanisms to solve this bandwidth saturation issues, in addition to allowing for communications with less latency (even real-time applications). In Section 2 we have also described these solutions, which consist of deploying small-scale clouds or servers on the edge of the network. Unfortunately, these solutions have not been implemented yet worldwide. Besides, not all mobile cloud applications have real-time execution as their priority, and most importantly, many companies lack enough equity to cope with the expense of deploying small clouds at multiple base stations or other locations at the edge of the network.

For all these reasons, we propose a new Mobile Cloud Computing (MCC) model that, unlike the existing solutions, can be applied to the current clouds without substantial disbursement. Our solution involves groups of volunteer users forming virtual platforms that act as resources to one or more clouds. Apart from cost-savings, the goals of our proposed model are:

- Elasticity: a cloud system that uses our solution can use the computing resources provided by the volunteer platforms whenever needed, enabling the system to adapt to significant workload changes. With our solution, a mobile cloud application can have many volunteers subscribed and can adapt the execution of the mobile clouds application in a elastic way.
- Scalability: after all, the volunteer platforms provide an extension to the cloud computing and storage capabilities, so cloud systems that use our proposed model would allow more users to access their resources. As we demonstrate in the evaluation section, the performance with our solution improves the scalability of the system when the number of mobile users increase.
- Efficiency: in some cases, mobile users would rather access a device from a volunteer platform than from a remote cloud server (geographical proximity means fewer hops), thus reducing latency.
- Load balancing: as we explain later, the cloud controllers process the user requests and provide the mobile users with the corresponding cloud services, either by their own clouds or by devices from the volunteer platforms that collaborate with the cloud system. This scenario allows for the implementation of various load balancing schemes not to saturate the cloud. This features is not analyzed in this work as is provided as future work.

- Easy deployment: the clouds and the devices from the volunteer platforms must run an open-source BOINC server and client software, respectively, so this solution does not require significant alterations of the cloud infrastructure. We describe this deployment in section 3.6.

3.3. Architecture

The architecture of our proposed model is shown in Figure 5, which is a variation of the MCC basic architecture presented in Section 2.

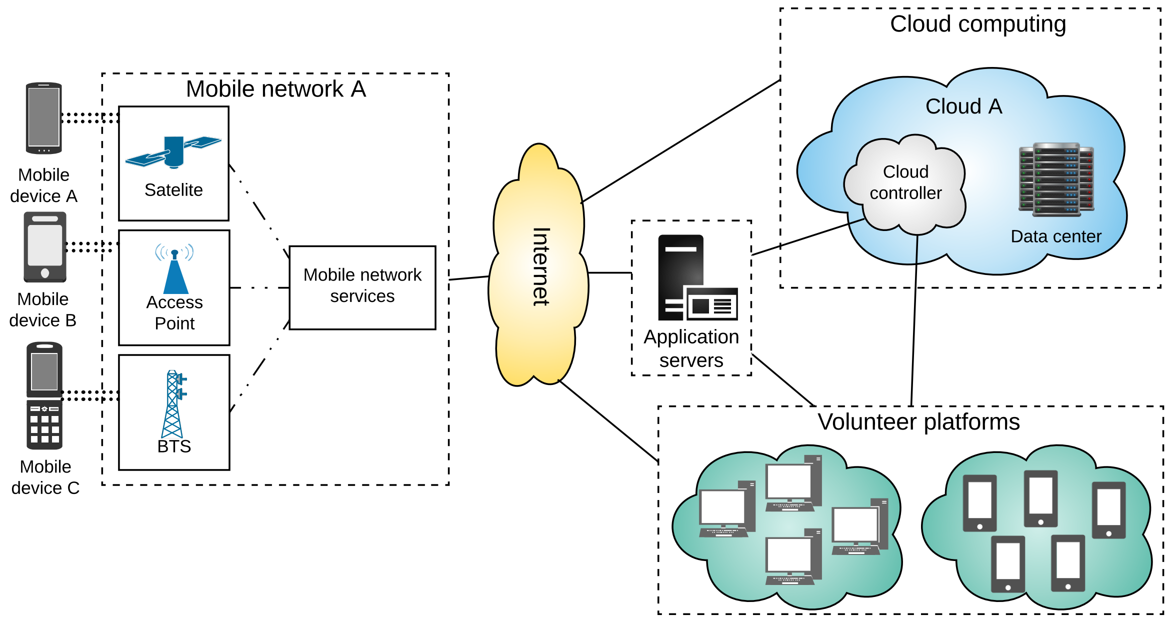


Figure 5: Architecture of our proposed model, based on the utilization of volunteer platforms.

The novel part of this approach is the utilization of volunteer platforms. A volunteer platform consists of multiple participating devices that want to donate their idle computing and storage resources to cloud systems, in a similar way to the millions of devices that currently contribute to BOINC scientific projects. A participating device that wants to contribute to a cloud system should download a variation of the BOINC open-source software [52] (available on Docker container [3] and Virtual Machine [4]), which executes in the idle CPU periods of the device, and should request work to the clouds that the device collaborates with. By the time a cloud system has the collaboration of multiple participating devices, it can distribute the devices in logical volunteer platforms or even define hierarchies, depending on their capabilities. For example, the volunteer platforms can be defined based on the storage capacity of the participating nodes, so that when a mobile user requests storage of a file to a cloud application, the cloud system should

[3]`https://boinc.berkeley.edu/trac/wiki/BoincDocker`
[4]`https://boinc.berkeley.edu/trac/wiki/VmServer`

replicate this file in a number of cloud servers and participating devices (from a volunteer platform) that are able to store a file of such size.

Each mobile user application that wants to use a cloud service should access the cloud system in an ordinary way (via the Internet). The cloud controller is then responsible for dealing with the user application request and providing the mobile user with the requested service. Nevertheless, in this model there are two options: providing the services using (1) the cloud servers of the system or (2) the volunteer resources of some participating devices (see Algorithm 1b [5]). From the point of view of a device that wants to donate resources to cloud services, it is necessary that it first subscribe to a cloud system as a participating device. Then, the cloud system would run some benchmarks on the participating device in order to test its capabilities. Once this has been done and depending on the type of service, the participating device should ask the cloud for tasks during its idle CPU time (see Algorithm 1a).

```
1: procedure SUBSCRIBE(srvc)    ▷ Subscription of device into cloud service srvc
2:     if not subscribed then    ▷ list is not empty
3:         send subscription request to srvc
4:         benchmarks ← receive answer from cloud    ▷ the cloud sends the benchmarks in order to know the capabilities of the device
5:         res ← execute benchmarks
6:         device_info_file ← create response file    ▷ this file should contain the benchmark results (res) and all other device information required (CPU model, RAM, GPS location, etc.)
7:         send device_info_file to the cloud
8:         url ← receive URL from cloud    ▷ this URL has the code the participating device should execute in order to collaborate in the service (e.g., a code that is able to receive computation requests and execute a neural network for a music identification service)
9:         code ← download code from url
10:        subscribed ← true
11:    end if
12:    run code in background
13: end procedure
```

(a)

```
1: procedure EXECUTE(tsk)    ▷ Remote execution of task tsk (e.g., a recorded audio)
2:     send request to cloud
3:     list ← receive answer from cloud    ▷ list of participating devices that are able to process the task; if the list is empty, that means the task should be executed by the cloud
4:     if list then    ▷ list is not empty
5:         err ← send tsk to N participating devices
6:         if not err then    ▷ there is no error
7:             res_list ← receive answers from the N participating devices
8:             res, err ← verify res_list    ▷ check if the quorum is reached
9:         end if
10:    end if
11:    if err or not list then    ▷ list is empty or there was an error related to the participating devices
12:        send tsk to cloud
13:        res ← receive answer from the cloud
14:    end if
15:    return res    ▷ computational result of tsk (e.g., identification that the short audio stored in tks corresponds to the song X)
16: end procedure
```

(b)

Algorithm 1: Examples of: (a) subscription of a participating device in a cloud service; (b) remote execution of a mobile user task.

[5]BOINC provides a form of redundant computing in which each computation is performed on multiple clients, the results are compared, and are accepted only when a consensus is reached. In some cases, new results must be created and sent. In Algorithm 1b, N is the replication factor, and the cloud administrators should choose it.

3.4. Volunteer Platforms

Volunteer platforms consist of groups of multiple participating devices with similar computing capabilities (decision of the company). As participating devices are going to run the BOINC client software, there are basically desktop computers and mobile devices. Since the participating devices are going to process tasks or store data of the mobile users, it is important to exercise caution over the battery lifetime for mobile devices. Fortunately, the BOINC client software for mobile devices computes only under the following conditions (as we mentioned in Section 2.3):

- The mobile device is plugged into a power source (AC or USB).
- The battery is over 90% of charge.
- The screen is off.

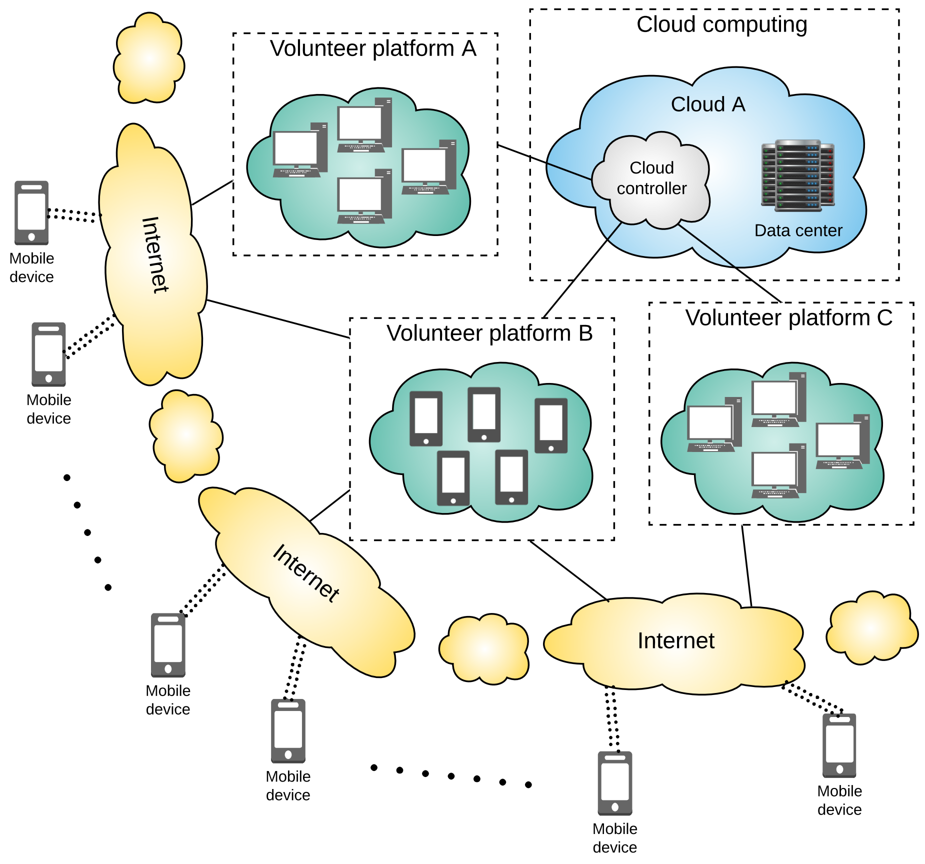


Figure 6: Mobile users access participating devices from volunteer platforms that are closer and are able to process the tasks needed.

In this way, the cloud tasks will not significantly reduce the battery life or the recharge time. For instance, an anonymous user can collaborate with a cloud system by just plugging their mobile volunteer device into a power source before going to sleep. Hence, the mobile volunteer device can participate in a cloud service while its owner is sleeping. Moreover, the ideal of this model is that mobile users leverage the computing and storage idle resources of volunteer devices that are geographically closer than the cloud remote servers, thereby preventing saturation of cloud networks and servers and also bypassing latency issues (because participating devices may be much nearer than the remote servers, so fewer hops are needed in order to arrive at the destination), as Figure 6 shows. However, as the resources provided by the participating nodes are volunteered, there is no assurance that these resources are going to be long-lasting. We can just say that they are 'volatile' resources and that the availability of participating devices is therefore vitally important. That is why our solution does not consist exclusively of volunteer platforms. The main processing and storage resources would be the ones provided by the cloud infrastructure in order to allow fault tolerance of the participating devices and therefore data loss. That said, the volunteer platforms will provide lots of benefits because they can back up files in storage services, process tasks, etc. even when there are no more available resources in the cloud. In other words, this solution does not change the current behavior of cloud services; it only provides more (inexpensive) resources to them and reduces the workload of the cloud.

3.5. Application Scenarios

Our proposed solution can be applied to different scenarios, among which we highlight storage and computing services.

3.5.1. Storage Services

Our solution, which consists of the usage of volunteer platforms as resource providers, can be applied to typical storage mobile cloud services [53], such as Dropbox, Google Drive, or OneDrive. In this scenario, once a participating device has subscribed to the cloud service when a mobile user wants to upload a file to the cloud, it sends the file to the cloud (for simplicity, we are ignoring all the protocol matters of these kinds of services). Then the file is stored in a number of cloud nodes (depending on the replication factor of the storage system), and then the encrypted file is sent to a number of participating devices of one or more volunteer platforms. In this way, each file is backed up in several places (for example, in two cloud servers and in two participating devices) so that the mobile user can download the file from both the cloud servers or the participating devices (for instance, based on proximity), and then verify its integrity by checking the hash against the cloud. This behavior is shown in Figure 7.

In addition, as the files stored by the participating devices are encrypted (e.g., using AES-256 [54]), there are no security risks in untrusted users storing private information, since the participating devices cannot access the file contents. Finally, we also assume that the mobile users can specify the maximum storage they want to donate. For example, the default value can be a 5% of the total storage capacity of the device (e.g., 25 GB for a computer with a hard disk of 500 GB).

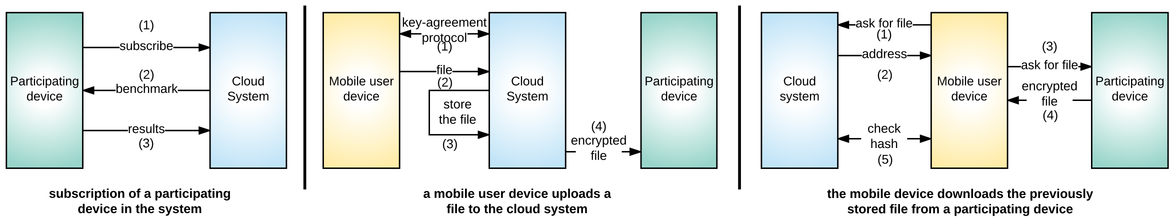


Figure 7: Example of a storage scenario.

3.5.2. Processing Services

Our model allows for the execution of multiple processing services. Music identification services (e.g., Shazam or ACRCloud) or optical character recognition (OCR) services are examples of this kind of processing services. In these scenarios, a mobile device sends a task (an audio file or a picture) to a remote cloud where the data is processed (identifying the song from the audio or recognizing a text from the picture) and the results of the computation performed are sent back to the mobile device. Without our solution, all the processing is performed within the cloud infrastructure. With our model, the processing task should be performed by the participating devices, thus reducing the load in the cloud. In our approach, when a participating device subscribes to a cloud service, it downloads from the cloud the application that it needs to execute (e.g., the binaries with the algorithms or the neural network to use). A mobile user device that wants to process some data first sends the processing request to the cloud system, which answers with a list of addresses of the participating devices (usually the addresses of all the devices of the same volunteer platform). Then, the user sends the task to a number of participating devices (two or more) in order to rely on the results of untrusted users. If the replies received from the participating devices match, the result is considered correct. This behavior is shown in Figure 8.

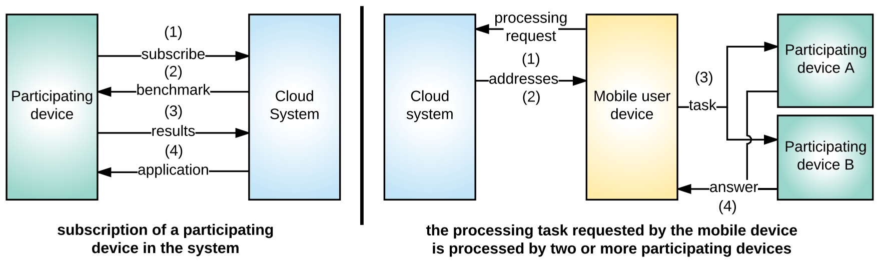


Figure 8: Example of a processing scenario.

In contrast to storage services, where the participating devices receive and store encrypted files, in processing services the computation tasks may be performed by untrusted users (the participating devices) over unencrypted data, so, in order to avoid security risks, it is compulsory that the participating devices only receive public content, such as street pictures or music audios that the user wants to identify. There are some novel techniques that try to perform computation over encrypted data [55], so probably in the future, our processing model can be applied also to tasks that use private information.

3.6. Deployment and Mobile Application Adaptation

This section describes the deployment of our solution for a typical application. This deployment is based on the BOINC behavior described in Section 2.3.1.

The first step is to transform the cloud application into a BOINC project. This step does not require changes in the BOINC software, since only some transformations in the cloud application are needed. In this transformation, we obtain two BOINC applications: one for the participating devices and other for the mobile user. Each application consists of a program and a set of workunits and results. The BOINC servers receive two types of requests:

- Requests from the mobile users.
- Requests from the participating devices. The server also stores the addresses of the different participating devices.

From the BOINC point of view, there are two types of clients: participating devices and mobile users. Both execute different applications. When a mobile user wants to execute a mobile cloud application, it sends the request to the server and obtains a new workunit. This workunit only includes a list containing the addresses of the participating devices. Then the mobile device selects one of these addresses (as shown in Algorithm 1) and sends the data to the selected participating device. After that, the participating device processes the application and returns the result to the mobile device, without the intervention of the BOINC server, lowering in this way the cloud load.

In order to avoid possible bottlenecks, several BOINC servers can be deployed similarly to other BOINC projects, like the SETI@Home project. Moreover, we can deploy several BOINC servers in different clouds, to reduce bottlenecks.

As other BOINC projects, all services needed for the application are installed on the device when it subscribes to the system. BOINC uses virtualization to allow the execution of applications in different hardware or operating system. The virtualization solution used by BOINC is VirtualBox, which is free and multiplatform. In this case, the recommended BOINC installer for Windows includes VirtualBox as well, and this is transparent for the user that installs BOINC. Similarly, the BOINC client installer can provide the installation of Docker in a transparent way to the users, simplifying its deployment on the volunteer nodes.

From the installation point of view, we have evaluated the time for installing BOINC in a typical project on several desktop computers and smartphones, and subscribing them to different projects. On the one hand, the installation process took an average of 30 seconds for both the desktop computers and the smartphones. On the other hand, the subscription phase took less than 10 seconds for all devices.

3.7. Incentive Scheme

Why would anonymous users want to donate their resources to cloud services? In BOINC, users donate their idle processing and storage resources to contribute to scientific projects, such as Climateprediction.net[6], that helps fight climate change; Rosetta@Home[7], that helps to find the cure for cancer and Alzheimer's; or SETI@Home[8], that helps to find extraterrestrial intelligence. However, this is not enough; that is why BOINC has an incentive scheme based on credits. BOINC projects grant credit to users to encourage the volunteer users to contribute to the system. Credit has no monetary value; it is only a measure of how much a volunteer has contributed to a project (credits are calculated from the floating point operations that a device has computed) [56]. In our solution, companies and organizations should also include an incentive scheme based on credits in their services, as BOINC does. In this way, volunteer users would be rewarded by their contribution to the mobile cloud computing services they collaborate with.

Apart from that, the enterprises that want to deploy our model can also reward the volunteer users with some 'special' functionalities. For example, a company that offers storage services to their mobile clients could grant the volunteer users with some premium features or even a professional account of one of their mobile applications for free.

3.8. Security Aspects

BOINC allows the project designers to use Secure Socket Layer (SSL) in their projects, so HTTPS (port 443) can be used in the log-in processes. Besides, BOINC uses the ports 31416 and 1043 to exchange data, so the client has to unblock them if they are behind a firewall. Similarly, the implementation of our approach must use specific ports that should be unblocked from the firewall to manage the access between clients. We propose two alternatives to ensure secure communication between the mobile users and the participating devices:

- Transport Layer Security (TLS, the last version is 1.2) [57]: it is available to most TCP applications (e.g., FTPS, SMTPS, and HTTPS).
- Simple Object Access Protocol (SOAP, last version is 1.2) [58]: it is a protocol for exchanging data using XML files. It can be combined with WS-Security (last version is WS-Security 1.1) [59] in order to add security. WS-Security is a protocol that guarantees authentication, confidentiality, and integrity of the data exchanged.

As we explained in Section 3.5.1 (see Figure 7), when a mobile user wants to upload a file using a storage service, it first has to specify an encryption key with the cloud through a key-agreement protocol.

[6]`http://www.climateprediction.net/`
[7]`https://boinc.bakerlab.org/`
[8]`https://setiathome.berkeley.edu/`

For that reason, we propose the Diffie Hellman Ephemeral (DHE) or the Elliptic Curve Diffie Hellman Ephemeral (ECDHE) [60] key-agreement protocols because they ensure the Perfect Forward Secrecy [61]. Then, encryption key should be stored in a secure local keystore by both the mobile device and the cloud. Besides, the file should be transmitted from the mobile device to the cloud via a secure channel (e.g., TLS), and then the file should be encrypted in the cloud side in order to offload the computation from the mobile device. A good option is to use a symmetric-key algorithm, such as AES256 [54] or 3DES [62]. Once the file is encrypted, the cloud can send it to multiple participating devices ensuring confidentiality. When the mobile device downloads the encrypted file from a participating device, it just has to verify the file hash with the cloud (to check integrity) and decrypt it using the encryption key previously stored in its keystore.

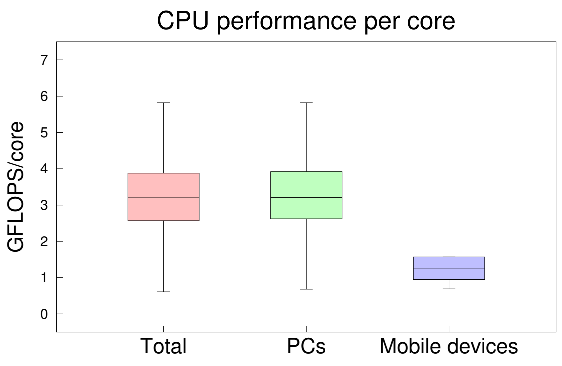
Apart from that, when a mobile user wants to execute a task, the cloud can reply to the mobile user with the list of participating devices that are able to execute the task. Exactly as BOINC works, the mobile user has to send the task to N different users, and, after receiving the computation results from all of them, check if the quorum is reached. For instance, suppose that a mobile user wants to apply an OCR program over a text in a poster, N is 3 and quorum is 2, so the user first takes a picture of the text, then requests to process this text to the cloud service, so the cloud replies with the list of participating devices that are able to process the task (normally, a whole volunteer platform). Then, the mobile user application sends the picture to three different participating devices (N value) and then it checks if at least two of the answers (quorum value) match. If the quorum is reached (e.g., two of the participating devices answer "Mr. Bean Street"), the result is considered to be correct. Otherwise, the mobile user requests it directly to the cloud. This behavior is also shown in Algorithm 1b. Apart from that, as described in [63], BOINC prevents to distribute malware among the volunteer computers because applications have only access to their own input and output files via sandboxing. Besides, the BOINC software is also able to use virtualization support [64], which would facilitate the deployment of our proposed model.

4. Evaluation

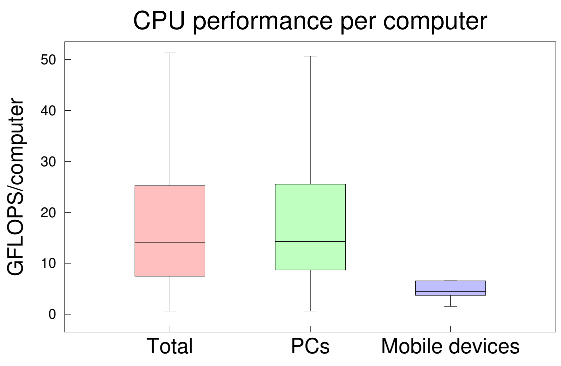
In this section we present the evaluation performed. In Section 4.1 we detail an analysis of the volunteer devices that participate in the famous SETI@Home project, apart from the description of how we managed to characterize three different individual devices. We have used these results in order to perform the experiments presented in Section 4.2, that consist of different case studies we have analyzed through realistic simulations.

4.1. Devices characterization

We have analyzed the CPU performance of the 138,252 computers of the SETI@Home project that were active on June 12, 2017, 22:02:19 UCT (published in [65]). After analyzing all the CPU models, we found that 134,182 (97.06%) of the total number of devices were desktop computers and laptops, while the remaining 4,070 (2.94%) computers were mobile devices. Figure 9 shows the CPU performance (GigaFLOPS/core or GigaFLOPS/computer) of the aforementioned SETI@Home volunteer devices. This huge difference (3.13 over 17.5 GigaFLOPS) between the performance per core (Figure 9a) and per computer (Figure 9b) is because the SETI@Home tasks use the maximum number of cores available for computation, ranging from 1 to 102 cores. As can be seen in the figure, mobile devices are much less powerful than the desktop and laptop computers on average (4.46 vs 17.91 GigaFLOPS/computer). We have used these SETI@Home CPU traces to model the power of the participating devices that form the volunteer platforms of the simulations presented in Section 4.2.



(a)



(b)

Figure 9: CPU performance of the volunteer computers of the SETI@Home project: (a) GFLOPS/core, (b) GFLOPS/computer.

In order to model the availability of the participating devices, we used the results obtained in [66]. This research analyzed about 230,000 availability traces obtained from the volunteer computers that participate in the SETI@Home project. According to this paper, 21% of the volunteer computers exhibit truly random availability intervals, and it also measured the goodness of fit of the resulting distributions using standard probability-probability (PP) plots. For availability, the authors noted that in most cases the Weibull distribution is a good fit. For unavailability, the distribution that offers the best fit is the log-normal. The parameters used for the Weibull distribution are $shape = 0.393$ and $scale = 2.964$. For the log-normal, the parameters obtained and used in ComBoS are a distribution with mean $\mu = -0.586$ and standard deviation $\sigma = 2.844$. All these parameters were obtained from [66] too. The availability and unavailability modeling, allow us to simulate the entrance and leaving of volunteer resources in the system.

Furthermore, because the software the participating devices in our proposed model is based on a small variation of the BOINC client software, we are also interested in evaluating the performance of individual devices participating in a real BOINC volunteer computing project. To make this possible, we have used the following devices:

- Desktop computer: Intel®Core^{TM} i7-4790 (4 cores (8 threads), 3.60 GHz), OS: Ubuntu 16.04.2 LTS, 8 GB of RAM memory.
- Mobile device: Woxter Zielo ZX840HD (8 cores, 1.7 GHz), OS: Android 4.4.2, 2 GB of RAM memory.
- ARM device: ODROID-C2 (4 cores, 1.5 GHz), OS: Ubuntu 16.04.2 LTS, 2 GB of RAM memory.

Each device has collaborated in the most famous BOINC project: the SETI@Home project. The results obtained are shown in Table 2, and demonstrate that the desktop computers currently provide more computational power to volunteer computing than the other kind of devices that participate in these projects.

Table 2: Computational results (GigaFLOPs executed in 2 days of uninterrupted computation) of the three devices after collaborating in the SETI@Home project.

Project	Desktop computer	Mobile device	ARM device
SETI@home	3,628,800	345,600	322,600

4.2. Case Studies

We have evaluated three different mobile cloud computing services as case studies: a generic processing service, a storage service, and a speech recognition processing service. In terms of evaluation, we have used ComBoS [67], a complete BOINC simulator created by the authors as a previous work, as a starting point. ComBoS is a public source software[9] and was implemented in C programming language, with the help of the tools provided by the MSG API of SimGrid [68] and is able to perform realistic simulations of the whole BOINC infrastructure, considering all its features: projects, servers, network, redundant computing, scheduling, etc. In order to evaluate both case studies, we have modified ComBoS to evaluate the scenario shown in Figure 10. This scenario consists of two groups of mobile devices, that access a cloud in order to use the services. It also has four volunteer platforms that provide computing and storage resources to the cloud. The bandwidth and latency values of the networks that connect the different components are also specified in Figure 10. All other parameters relevant to the simulations (number of devices of each type, power, etc.) are specified in each case study.

Table 3 shows the details of the platform used to simulate the case studies. Every execution in this section has simulated 100 hours. In order to account for the randomness of the simulations and to deem

[9]ComBoS can be downloaded from: `https://github.com/arcos-combos/combos`

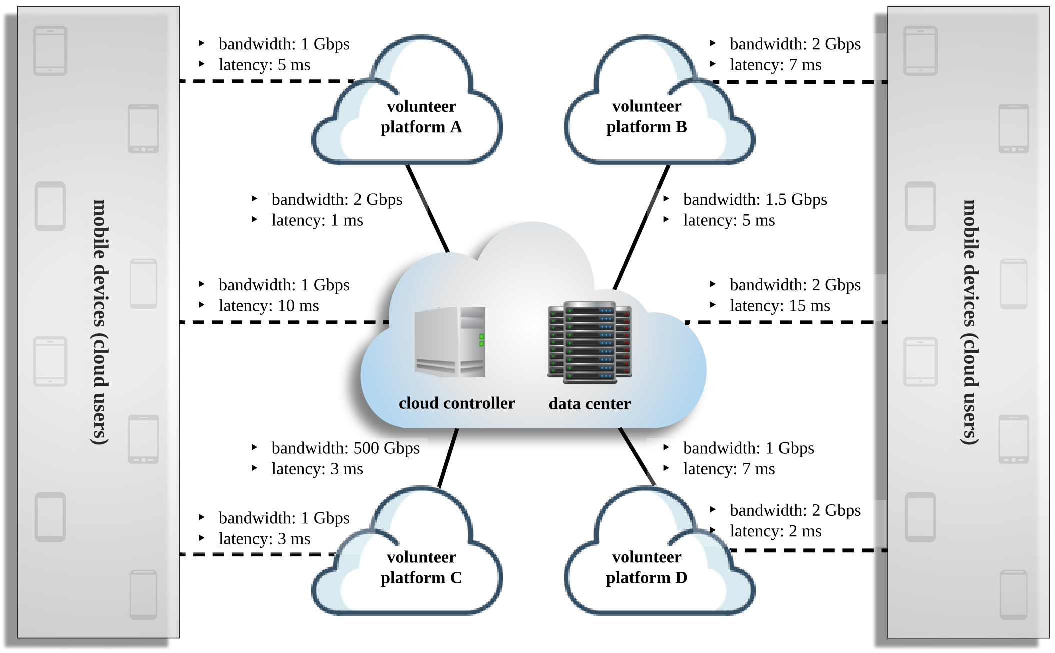


Figure 10: Scenario simulated in the experiments.

Table 3: Platform used in the evaluation.

	Value
Processor	Intel®Core^{TM} i7-920 (4 cores (8 threads), 2.67GHz)
RAM	32 GB
Operating System	Ubuntu 14.04.5 LTS
Kernel	3.13.0-119-generic
SimGrid version	3.11

the results reliable, each simulation result presented in this section is based on the average of 20 runs. For a 95% confidence interval, the error is less than ± 2% for all values.

4.2.1. Case study 1: Processing Service

A good case study to evaluate our proposed model is to analyze its performance of processing services. These processing services can range from a music identification service (e.g., Shazam) to a text recognition service (e.g., an OCR). We considered the scenario shown in Figure 10, where each volunteer platform has 250 participating devices, and there are from 20,000 to 100,000 mobile users. The cloud infrastructure consists of 20 nodes with a computing power of 50 GigaFLOPS each, and each mobile device requests the cloud to compute a task of 20 GigaFLOPs[10] (based on the results of [69]) and 5 MB every 30 minutes on average. We have considered three different configurations:

535 • Configuration 1: it corresponds to the original behavior of a cloud system (without using volunteer platforms). All the processing tasks are performed in the cloud.

• Configuration 2: both volunteer platforms are formed by participating devices in the same proportion and with the same properties (power and availability) as in the SETI@Home project (see Section 4.1).

540 • Configuration 3: both volunteer platforms are formed only by mobile devices - the participating devices are only mobile devices in this configuration - with the same properties (power and availability) as in the SETI@Home project (see Section 4.1).

In configurations 2 and 3, the tasks are computed either by the cloud or by the participating devices on a round-robin basis. In the case a task is computed by a volunteer platform instead of by the cloud, 545 three different participating devices should compute the task with a quorum of two.

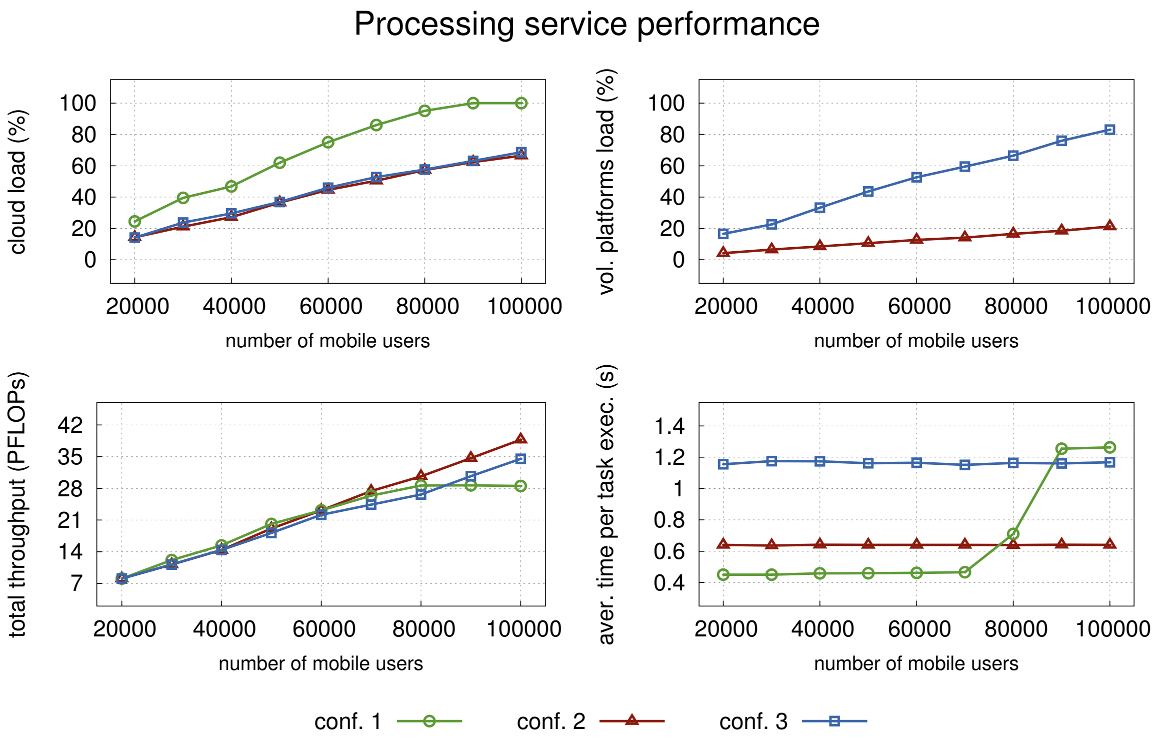


Figure 11: Case study 1: performance of the processing service for the three different configurations.

Figure 11 shows the results of this experiment: the load[11] of both the cloud and the volunteer

[10] We distinguish between FLOPS (floating point operations per second) and FLOPs (floating point operations).
[11] We considered the load as the maximum of the network and the CPU load.

platforms, the total throughput of the system in PetaFLOPs, and the average time a task is executed. With configuration 1, the cloud became saturated with almost 85,000 mobile users. By contrast, this did not happen with configurations 2 and 3, because the computation of the tasks is shared by both the participating devices and the cloud, not only by the cloud as in the previous configuration. As it is shown in the figure, the use of volunteer platforms allows for an increase in the scalability and in the total throughput of the system, since more users can process their tasks in the system. Finally, although the average time per task execution in configuration 1 is less than in the rest of configurations (except when the cloud is saturated), this difference is not significant, especially for configuration 2 (less than 200 ms), which shows that our approach would not have a negative impact on the user experience.

4.2.2. Case study 2: Storage Service

This second case study is about a file storage service. The scenario is the same as in the previous case (Figure 10), where each volunteer platform has 25.000 participating devices. The cloud infrastructure consists of 200 nodes, each with 3.2 Terabytes, making a total of 640 Terabytes of storage. In this case study there are 1 million cloud users and each one uploads an average of 4.5 files of 50 MB each (following an exponential distribution) to the cloud service. We have again considered three different configurations:

- Configuration 1: it corresponds to the original behavior of a cloud system (without using volunteer platforms). All the files are downloaded from the cloud. Each file should be replicated three times in the cloud servers.
- Configuration 2: both volunteer platforms are formed by participating devices with a storage capacity that follows a statistical normal distribution, with $\mu = 5$ and $\sigma = 0.75$ (average 5 GB per device). Each file should be replicated two times in the cloud servers and two times in the participating devices.
- Configuration 3: both volunteer platforms are formed by participating devices with a storage capacity that follows an statistical normal distribution, with $\mu = 10$ and $\sigma = 1$ (average 10 GB per device). Each file should be replicated two times in the cloud servers and two times in the participating devices.

Figure 12 shows the results of this experiment. As it can be seen in graphs (a) and (b), with the first configuration (without using volunteer platforms), the cloud is not able to store more files because there is no more available space in their nodes. On the other hand, with configurations 2 and 3, the cloud is not saturated, and the service is then able to store and back up all the files from the 1 million users thanks to the storage resources donated by the participating users. Moreover, in graph (c) we show the average time required to download a file by the mobile users, assuming the mobile users download a file every 2 hours on average. With configuration 1, the cloud network becomes saturated soon; that is why

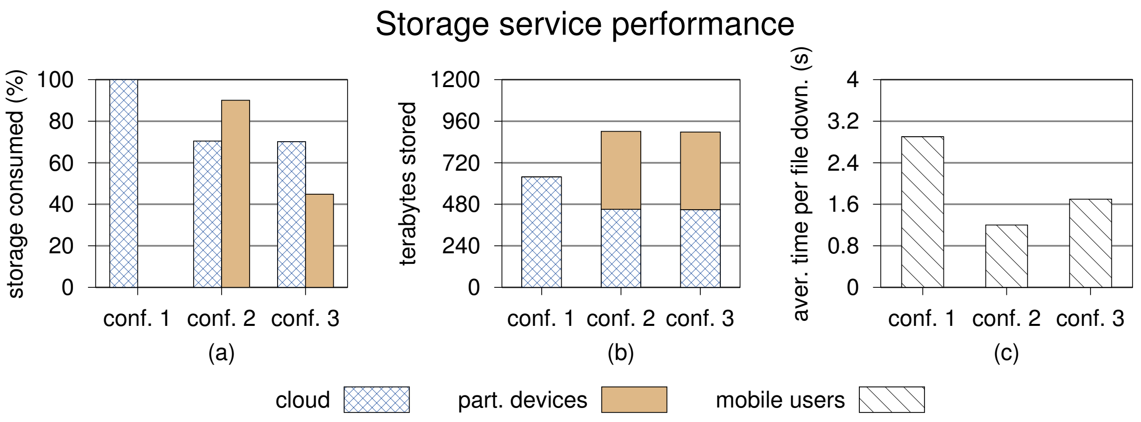


Figure 12: Case study 2: performance of the storage service for the three different configurations.

the download time is higher than in configurations 2 and 3, where the mobile users also download files from the volunteer platforms.

4.2.3. Case study 3: Speech Recognition

In [70], the authors show that the new multimedia applications are changing the current cloud computing architecture. They analyze different multimedia applications, such as face and speech recognition, or mobile augmented reality. In the paper, the authors made all the experiments using a Dell Latitude 2102 as the unique mobile device. However, in this section, we have simulated their speech recognition application considering the scenario of Figure 10, and using the same configurations as in the first case study (Processing Service, Section 4.2.1). Each volunteer platform has 250 participating devices, and there are from 20,000 to 200,000 mobile users. The speech recognition application is based on an open-source speech-to-text framework that uses Hidden Markov Models (HMM) recognition systems [71]. In this application, the average request size is 243 KB, and the cloud response size is 60 bytes on average. The request and response size and the rest of the parameters are obtained from [70] too. We have simulated cases in where 1 to 22 words are recognized.

Figure 13 shows the results of this last experiment, considering the cloud and volunteer platforms load, the total throughput, and the average time a task is executed (similarly as in the first case study). With configuration 1, the cloud became saturated with almost 120,000 mobile users. As in the generic processing service, this did not happen with configurations 2 and 3. As it is shown in the figure, the use of volunteer platforms allows for an increase in the scalability and in the total throughput of the system, since more users can process their tasks in the system. Finally, although the average time per task execution in configuration 1 is less than in the rest of configurations (except when the cloud is saturated), this difference is not significant, especially for configuration 2 (less than 50 ms), which shows that our approach would not have a negative impact on the user experience. The participating devices

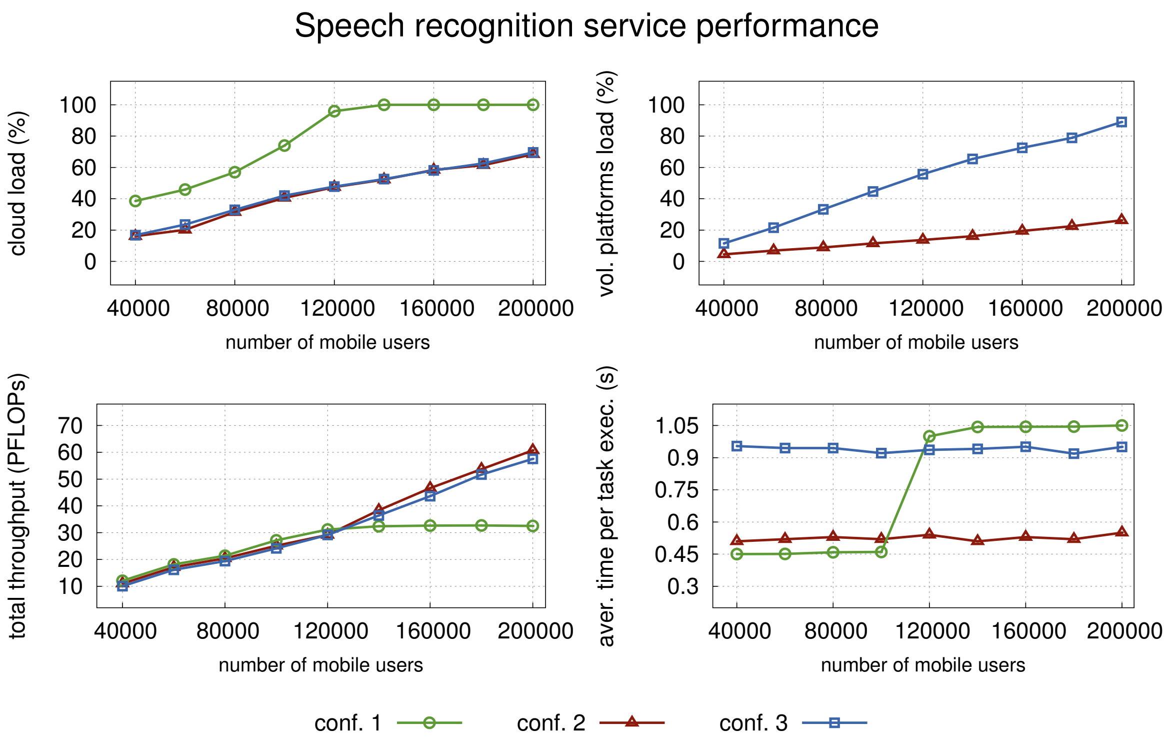


Figure 13: Case study 3: performance of the speech recognition service for the three different configurations.

allow avoiding a performance loss when a high number of clients consume the services, also improving the availability and the usability of the service.

5. Conclusion and Future Work

This paper gave an overview of mobile cloud computing (MCC), in addition to a new MCC model that can provide more computing and storage resources to public, private or hybrid clouds. The proposed heterogeneous model uses the computing and storage resources of devices from the general public to contribute to cloud systems, so the organizations can leverage the idle periods of these devices to gain computing and storage resources for their cloud services, in a similar way that volunteer devices contribute to BOINC projects. As we have shown throughout the paper, our proposed model can provide several benefits to the cloud systems, including cost savings, as it avoids monetary investments in infrastructure, since the resources are volunteered; elasticity, as it enables the system to adapt to significant workload changes in the cloud just by using the volunteered resources; scalability, as it provides more computing and storage resources to the system, so more users can use these resources; efficiency, as the volunteer devices can be closer than the cloud servers for the mobile users, thus reducing the network latency; load balancing, as the cloud controllers can choose to use the cloud's own resources or the volunteer resources, so different load balancing algorithms can be implemented; easy deployment, as we propose to use the current BOINC open-source software in order to deploy our solution in current cloud systems. Moreover, with the evaluation performed we have also shown that our proposed model is a feasible solution for cloud services that have a large number of mobile users. This evaluation demonstrate the scalability of our solution.

For future work, we plan to use the BOINC software to deploy a prototype of this approach. We look forward to analyzing the impact of the proposed model in different scenarios, not only the ones showed in this document. Moreover, we want to analyze different load balancing policies, in order to increase the performance and scalability.

Acknowledgements

This work has been partially supported by the Spanish MINISTERIO DE ECONOMÍA Y COMPETITIVIDAD under the project grant TIN2016-79637-P TOWARDS UNIFICATION OF HPC AND BIG DATA PARADIGMS.

References

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, M. Zaharia, A view of cloud computing, Commun. ACM 53 (4) (2010) 50–58. doi:10.1145/1721654.1721672.
URL http://doi.acm.org/10.1145/1721654.1721672

[2] H. T. Dinh, C. Lee, D. Niyato, P. Wang, A survey of mobile cloud computing: architecture, applications, and approaches, Wireless Communications and Mobile Computing 13 (18) (2013) 1587–1611. doi:10.1002/wcm.1203.
URL http://dx.doi.org/10.1002/wcm.1203

[3] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, V. Young, Mobile edge computing—a key technology towards 5g, ETSI White Paper 11.

[4] T. X. Tran, A. Hajisami, P. Pandey, D. Pompili, Collaborative mobile edge computing in 5g networks: New paradigms, scenarios, and challenges, IEEE Communications Magazine 55 (4) (2017) 54–61. doi:10.1109/MCOM.2017.1600863.

[5] S. Gilbert, N. Lynch, Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services, SIGACT News 33 (2) (2002) 51–59. doi:10.1145/564585.564601.
URL http://doi.acm.org/10.1145/564585.564601

[6] G. Lewis, S. Echeverría, S. Simanta, B. Bradshaw, J. Root, Tactical cloudlets: Moving cloud computing to the edge, in: 2014 IEEE Military Communications Conference, 2014, pp. 1440–1446. doi:10.1109/MILCOM.2014.238.

[7] F. R. Duro, J. G. Blas, D. Higuero, O. Perez, J. Carretero, Cosmic: A hierarchical cloudlet-based storage architecture for mobile clouds, Simulation Modelling Practice and Theory 50 (2015) 3 – 19, special Issue on Resource Management in Mobile Clouds. doi:http://dx.doi.org/10.1016/j.simpat.2014.07.007.
URL http://www.sciencedirect.com/science/article/pii/S1569190X1400118X

[8] I. Stojmenovic, S. Wen, The fog computing paradigm: Scenarios and security issues, in: 2014 Federated Conference on Computer Science and Information Systems, 2014, pp. 1–8. doi:10.15439/2014F503.

[9] S. Yi, C. Li, Q. Li, A survey of fog computing: Concepts, applications and issues, in: Proceedings of the 2015 Workshop on Mobile Big Data, Mobidata '15, ACM, New York, NY, USA, 2015, pp. 37–42. doi:10.1145/2757384.2757397.
URL http://doi.acm.org/10.1145/2757384.2757397

[10] P. M. Mell, T. Grance, Sp 800-145. the nist definition of cloud computing, Tech. rep., National Institute of Standards & Technology, Gaithersburg, MD, United States (2011).

[11] H. Qi, A. Gani, Research on mobile cloud computing: Review, trend and perspectives, in: Digital Information and Communication Technology and it's Applications (DICTAP), 2012 Second International Conference on, 2012, pp. 195–202. doi:10.1109/DICTAP.2012.6215350.

[12] A. u. R. Khan, M. Othman, S. A. Madani, S. U. Khan, A survey of mobile cloud computing application models, IEEE Communications Surveys Tutorials 16 (1) (2014) 393–413. doi:10.1109/SURV.2013.062613.00160.

[13] N. Fernando, S. W. Loke, W. Rahayu, Mobile cloud computing: A survey, Future Generation Computer Systems 29 (1) (2013) 84–106, including Special section: AIRCC-NetCoM 2009 and Special section: Clouds and Service-Oriented Architectures. doi:https://doi.org/10.1016/j.future.2012.05.023.
URL http://www.sciencedirect.com/science/article/pii/S0167739X12001318

[14] E. E. Marinelli, Hyrax: cloud computing on mobile devices using mapreduce, Tech. rep., DTIC Document (2009).

[15] S. Zachariadis, C. Mascolo, W. Emmerich, SATIN: A Component Model for Mobile Self Organisation, Springer Berlin Heidelberg, Berlin, Heidelberg, 2004, Ch. 1, pp. 1303–1321. doi:

10.1007/978-3-540-30469-2_31.
URL http://dx.doi.org/10.1007/978-3-540-30469-2_31

[16] L. M. Vaquero, L. Rodero-Merino, Finding your way in the fog: Towards a comprehensive definition of fog computing, SIGCOMM Comput. Commun. Rev. 44 (5) (2014) 27–32. doi:10.1145/2677046.2677052.
URL http://doi.acm.org/10.1145/2677046.2677052

[17] H. Gupta, A. V. Dastjerdi, S. K. Ghosh, R. Buyya, ifogsim: A toolkit for modeling and simulation of resource management techniques in internet of things, edge and fog computing environments, CoRR abs/1606.02007.
URL http://arxiv.org/abs/1606.02007

[18] F. Bonomi, R. Milito, P. Natarajan, J. Zhu, Fog Computing: A Platform for Internet of Things and Analytics, Vol. 546, Springer International Publishing, Cham, 2014, Ch. 1, pp. 169–186. doi:10.1007/978-3-319-05029-4_7.
URL http://dx.doi.org/10.1007/978-3-319-05029-4_7

[19] T. H. Luan, L. Gao, Z. Li, Y. Xiang, L. Sun, Fog computing: Focusing on mobile users at the edge, CoRR abs/1502.01815.
URL http://arxiv.org/abs/1502.01815

[20] E. Borcoci, Fog Computing, Mobile Edge Computing, Cloudlets - which one?, iaria.org, 2017, Ch. 1, pp. 1–122.
URL https://www.iaria.org/conferences2016/filesICSNC16/Softnet2016_Tutorial_Fog-MEC-Cloudlets-E.Borcoci-v1.1.pdf

[21] D. Evans, The internet of things: How the next evolution of the internet is changing everything, Whitepaper, CISCO Internet Business Solutions Group (IBSG) 1 (2011) 1–11.

[22] J. Gubbi, R. Buyya, S. Marusic, M. Palaniswami, Internet of things (iot): A vision, architectural elements, and future directions, Future generation computer systems 29 (7) (2013) 1645–1660, including Special sections: Cyber-enabled Distributed Computing for Ubiquitous Cloud and Network Services: Cloud Computing and Scientific Applications — Big Data, Scalable Analytics, and Beyond. doi:https://doi.org/10.1016/j.future.2013.01.010.
URL http://www.sciencedirect.com/science/article/pii/S0167739X13000241

[23] P. of the Third ACM Workshop on Mobile Cloud Computing, A. N. Y. N. U. Services, MCS '12 (Eds.), Cloudlets: bringing the cloud to the mobile user, 2012.

[24] A. Ceselli, M. Premoli, S. Secci, Mobile edge cloud network design optimization, IEEE/ACM Transactions on Networking PP (99) (2017) 1–14. doi:10.1109/TNET.2017.2652850.

[25] M. Satyanarayanan, P. Bahl, R. Caceres, N. Davies, The case for vm-based cloudlets in mobile computing, IEEE Pervasive Computing 8 (4) (2009) 14–23. doi:10.1109/MPRV.2009.82.

[26] K. Ha, M. Satyanarayanan, Openstack++ for cloudlet deployment, School of Computer Science Carnegie Mellon University Pittsburgh.

[27] A. Ahmed, E. Ahmed, A survey on mobile edge computing, in: 2016 10th International Conference on Intelligent Systems and Control (ISCO), 2016, pp. 1–8. doi:10.1109/ISCO.2016.7727082.

[28] T. Soyata, R. Muraleedharan, C. Funai, M. Kwon, W. Heinzelman, Cloud-vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture, in: 2012 IEEE Symposium on Computers and Communications (ISCC), 2012, pp. 000059–000066. doi:10.1109/ISCC.2012.6249269.

[29] R.-I. Ciobanu, C. Negru, F. Pop, C. Dobre, C. X. Mavromoustakis, G. Mastorakis, Drop computing: Ad-hoc dynamic collaborative computing, Future Generation Computer Systems.

[30] R.-C. Marin, Hybrid contextual cloud in ubiquitous platforms comprising of smartphones, International Journal of Intelligent Systems Technologies and Applications 12 (1) (2013) 4–17.

[31] B. Zhou, A. V. Dastjerdi, R. Calheiros, S. Srirama, R. Buyya, mcloud: A context-aware offloading framework for heterogeneous mobile cloud, IEEE Transactions on Services Computing.

[32] M. R. M. Assis, L. F. Bittencourt, R. Tolosana-Calasanz, Cloud federation: Characterisation and conceptual model, in: Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing, UCC '14, IEEE Computer Society, Washington, DC, USA, 2014, pp. 585–590. doi:10.1109/UCC.2014.90.
URL http://dx.doi.org/10.1109/UCC.2014.90

[33] S. Banerjee, Edge computing in the extreme and its applications, in: Proceedings of the Eighth Wireless of the Students, by the Students, and for the Students Workshop, S3, ACM, New York, NY, USA, 2016, pp. 2–2. doi:10.1145/2987354.2990474.
URL http://doi.acm.org/10.1145/2987354.2990474

[34] B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A. D. Joseph, R. Katz, S. Shenker, I. Stoica, Mesos: A platform for fine-grained resource sharing in the data center, in: Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation, NSDI'11, USENIX Association, Berkeley, CA, USA, 2011, pp. 295–308.
URL http://dl.acm.org/citation.cfm?id=1972457.1972488

[35] B. Burns, B. Grant, D. Oppenheimer, E. Brewer, J. Wilkes, Borg, omega, and kubernetes, Commun. ACM 59 (5) (2016) 50–57. doi:10.1145/2890784.
URL http://doi.acm.org/10.1145/2890784

[36] W. Minnebo, B. Van Houdt, Pull versus push mechanism in large distributed networks: Closed form results, in: Proceedings of the 24th International Teletraffic Congress, ITC '12, International Teletraffic Congress, 2012, pp. 9:1–9:8.
URL http://dl.acm.org/citation.cfm?id=2414276.2414288

[37] A. L. Beberg, V. S. Pande, D. L. Ensign, S. Khaliq, G. Jayachandran, Folding@home: Lessons from eight years of volunteer distributed computing, 2009 IEEE International Symposium on Parallel & Distributed Processing (IPDPS) 00 (2009) 1–8. doi:doi.ieeecomputersociety.org/10.1109/IPDPS.2009.5160922.

[38] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, I. Brandic, Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility, Future Generation Computer Systems 25 (6) (2009) 599 – 616. doi:https://doi.org/10.1016/j.future.2008.12.001.
URL http://www.sciencedirect.com/science/article/pii/S0167739X08001957

[39] B. Ismail, E. Mostajeran, M. Bazli Ab Karim, W. Ming Tat, S. Setapa, J.-Y. Luke, H. Ong, Evaluation of docker as edge computing platform, in: 2015 IEEE Conference on Open Systems (ICOS), 2015, pp. 130–135.

[40] BOINC, Boinc and docker, https://boinc.berkeley.edu/trac/wiki/BoincDocker, online ((accessed 02 November 2017, 22:02:19 UCT)).

[41] M. Yannuzzi, R. A. Milito, R. Serral-Gracià, D. Montero, M. Nemirovsky, Key ingredients in an iot recipe: Fog computing, cloud computing, and more fog computing, 2014 IEEE 19th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD) (2014) 325–329.

[42] L. F. G. Sarmenta, Volunteer computing, Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, aAI0803463 (2001).

[43] D. P. Anderson, Boinc: a system for public-resource computing and storage, in: Fifth IEEE/ACM International Workshop on Grid Computing, 2004, pp. 4–10. doi:10.1109/GRID.2004.14.

[44] R. Nakanishi, A study on utilization of TV sets in volunteer computing, in: ITE Tech. Rep., Vol. 40 of BCT '16, 2016, pp. 17–20.

[45] R. Cushing, G. H. H. Putra, S. Koulouzis, A. Belloum, M. Bubak, C. de Laat, Distributed computing on an ensemble of browsers, IEEE Internet Computing 17 (5) (2013) 54–61. doi:10.1109/MIC.2013.3.

[46] P. Czarnul, J. Kuchta, M. Matuszek, Parallel Computations in the Volunteer–Based Comcute System, Springer Berlin Heidelberg, Berlin, Heidelberg, 2014. doi:10.1007/978-3-642-55224-3_25.
URL http://dx.doi.org/10.1007/978-3-642-55224-3_25

[47] A. Calderón, F. García-Carballeira, B. Bergua, L. M. Sánchez, J. Carretero, Expanding the volunteer computing scenario: A novel approach to use parallel applications on volunteer computing, Future Generation Computer Systems 28 (6) (2012) 881 – 889, including Special sections SS: Volunteer Computing and Desktop Grids and SS: Mobile Ubiquitous Computing. doi:http://dx.doi.org/10.1016/j.future.2011.04.004.
URL http://www.sciencedirect.com/science/article/pii/S0167739X11000550

[48] J. Balicki, W. Korłub, J. Paluszak, Big Data Processing by Volunteer Computing Supported by Intelligent Agents, Springer International Publishing, Cham, 2015. doi:10.1007/978-3-319-19941-2_26.
URL http://dx.doi.org/10.1007/978-3-319-19941-2_26

[49] R. Bruno, P. Ferreira, freecycles: Efficient data distribution for volunteer computing, in: Proceedings of the Fourth International Workshop on Cloud Data and Platforms, CloudDP '14, ACM, New York, NY, USA, 2014, pp. 4:1–4:6. doi:10.1145/2592784.2592788.
URL http://doi.acm.org/10.1145/2592784.2592788

[50] V. D. Cunsolo, S. Distefano, A. Puliafito, M. Scarpa, Volunteer computing and desktop cloud: The cloud@home paradigm, in: 2009 Eighth IEEE International Symposium on Network Computing and Applications, 2009, pp. 134–139. doi:10.1109/NCA.2009.41.

[51] S. Alonso-Monsalve, F. García-Carballeira, A. Calderón, Fog computing through public-resource computing and storage, in: The 2nd International Conference on Fog and Mobile Edge Computing (FMEC 2017), IEEE, 2017, pp. 81–87. doi:10.1109/FMEC.2017.7946412.

[52] D. P. Anderson et al., Boinc, https://github.com/BOINC/boinc (2017).

[53] Y. Cui, Z. Lai, N. Dai, A first look at mobile cloud storage services: architecture, experimentation, and challenges, IEEE Network 30 (4) (2016) 16–21. doi:10.1109/MNET.2016.7513859.

[54] M. J. Dworkin, E. B. Barker, J. R. Nechvatal, J. Foti, L. E. Bassham, E. Roback, J. F. Dray Jr, Advanced encryption standard (aes), Federal Inf. Process. Stds.(NIST FIPS)-197.

[55] F.-H. Liu, Computation over encrypted data, Cloud Computing Security: Foundations and Challenges (2016) 305.

[56] D. P. Anderson, J. McLeod, Local scheduling for volunteer computing, in: 2007 IEEE International Parallel and Distributed Processing Symposium, 2007, pp. 1–8. doi:10.1109/IPDPS.2007.370667.

[57] T. Dierks, E. Rescorla, The transport layer security (tls) protocol version 1.2, in: IETF RFC 5246, 2008, pp. 1–19.

[58] M. Gudgin, M. Hadley, N. Mendelsohn, J.-J. Moreau, H. F. Nielsen, A. Karmarkar, Y. Lafon, Simple object access protocol (soap) 1.2, World Wide Web Consortium.

[59] A. Nadalin, C. Kaler, R. Monzillo, P. Hallam-Baker, Web services security: Soap message security 1.1 (ws-security 2004). oasis standard specification, 2006.

[60] S. Blake-Wilson, B. Moeller, V. Gupta, C. Hawk, N. Bolyard, Elliptic curve cryptography (ecc) cipher suites for transport layer security (tls) (2006).

[61] H. Krawczyk, Perfect Forward Secrecy, Springer US, Boston, MA, 2011. doi:10.1007/978-1-4419-5906-5_90.
URL http://dx.doi.org/10.1007/978-1-4419-5906-5_90

[62] W. C. Barker, E. Barker, U. D. of Commerce, N. I. of Standards, Technology, Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher: NIST Special Publication 800-67, Revision 2, CreateSpace Independent Publishing Platform, USA, 2012.

[63] D. P. Anderson, Volunteer computing: The ultimate cloud, Crossroads 16 (3) (2010) 7–10. doi:10.1145/1734160.1734164.
URL http://doi.acm.org/10.1145/1734160.1734164

[64] BOINC, Virtualbox, http://boinc.berkeley.edu/wiki/VirtualBox (2016).

[65] SETI@Home, Cpu performance, https://setiathome.berkeley.edu/cpu_list.php, online ((accessed 12 June 2017, 22:02:19 UCT)).

[66] B. Javadi, D. Kondo, J. M. Vincent, D. P. Anderson, Discovering statistical models of availability in large distributed systems: An empirical study of seti@home, IEEE Transactions on Parallel and Distributed Systems 22 (11) (2011) 1896–1903. doi:10.1109/TPDS.2011.50.

[67] S. Alonso-Monsalve, F. García-Carballeira, A. Calderón, Combos: A complete simulator of volunteer computing and desktop grids, Simulation Modelling Practice and Theory 77 (2017) 197 – 211. doi:https://doi.org/10.1016/j.simpat.2017.06.002.
URL http://www.sciencedirect.com/science/article/pii/S1569190X17301028

[68] H. Casanova, A. Giersch, A. Legrand, M. Quinson, F. Suter, Versatile, scalable, and accurate simulation of distributed applications and platforms, Journal of Parallel and Distributed Computing 74 (10) (2014) 2899–2917.
URL http://hal.inria.fr/hal-01017319

[69] B. Ramesh, A. Bhardwaj, J. Richardson, A. D. George, H. Lam, Optimization and evaluation of image- and signal-processing kernels on the ti c6678 multi-core dsp, in: 2014 IEEE High Performance Extreme Computing Conference (HPEC), 2014, pp. 1–6. doi:10.1109/HPEC.2014.7040989.

[70] K. Ha, P. Pillai, G. Lewis, S. Simanta, S. Clinch, N. Davies, M. Satyanarayanan, The impact of mobile multimedia applications on data center consolidation, in: 2013 IEEE International Conference on Cloud Engineering (IC2E), 2013, pp. 166–176. doi:10.1109/IC2E.2013.17.

[71] Sphinx-4, Sphinx-4: A speech recognizer written entirely in the Java programming language, http://cmusphinx.sourceforge.net/sphinx4/, online.

Saúl Alonso Monsalve received, with honors, his MS degree in Computer Science and Technology in 2017 at University Carlos III of Madrid. He is currently a PhD student at CERN. Previously, he was a teaching and research assistant in the Computer Science and Engineering Department at University Carlos III of Madrid. His research interests include, but are not limited to, high-performance computing, distributed systems, and deep learning. He is a collaborator member, among others, of the Deep Underground Neutrino Experiment (DUNE).

Félix García-Carballeira received the M.S. degree in Computer Science in 1993 at the Universidad Politécnica de Madrid, and the PhD degree in Computer Science in 1996 at the same university. From 1996 to 2000 was associate professor in the Department of Computer Architecture at the Universidad Politécnica de Madrid. He is currently a full professor in the Computer Science Department at the Universidad Carlos III de Madrid. His research interests include high performance computing and parallel and distributed file systems. He is co-author of 13 books and he has published more than 80 articles in journals and conferences.

Alejandro Calderón received his M.S. degree in Computer Science from the Universidad Politécnica de Madrid in 2000 and his Ph.D. in 2005 from the Universidad Carlos III de Madrid. He is an associate professor in the Computer Science Department at the Carlos III University of Madrid (Spain). His research interests include high performance computing, large distributed computing, and parallel file systems. Alejandro has participated in the implementation of MiMPI, a multithread implementation of MPI, and in the Expand parallel file system too. He is the coauthor of more than 40 articles in journals and conferences.

New mobile cloud computing model, in which part of the resources of the cloud are provided by platforms of volunteer devices.

Benefits in cost savings, elasticity, scalability, load balancing, and efficiency.

Simulation-based evaluation.