Volunteer Computing on Mobile Devices: State of the Art and Future Research Directions

Cristiano Tapparello*, Colin Funai*, Shurouq Hijazi*, Abner Aquino*,
Bora Karaouglu, He Ba*, Jiye Shi† and Wendi Heinzelman*

*Department of Electrical and Computer Engineering, University of Rochester, Rochester, NY, USA
†UCB Pharma, 216 Bath Road Slough, SL1 4EN, United Kingdom

Abstract

Different forms of parallel computing have been proposed to address the high computational requirements of many applications, following the principle that large computational problems can often be divided into smaller ones. Building on advances in parallel and distributed computing, volunteer computing has been shown to be an efficient way to exploit the computational resources of devices that are available around the world and that are under utilized for most of their time. The idea of including mobile devices, such as smartphones and tablets, in existing distributed volunteer computing systems has recently been investigated. In this chapter, we present the current state of the art in the mobile volunteer computing research field, where personal mobile devices are the elements that perform the computation. Starting from the motivations and challenges behind the adoption of personal mobile devices as computational resources, we then provide a literature review of the different architectures that have been proposed to support parallel and distributed computing and how these architectures have been adapted to use mobile devices for distributed computing. Finally, we present some open issues that need to be investigated in order to extend user participation and improve the overall system performance for mobile volunteer computing.

I. Introduction

In recent years, the computational requirements of various applications in domains ranging from healthcare to finance have increased dramatically. Several computing infrastructures have been proposed and, among them, parallel computing has been shown to be a viable solution to meeting this increasing computational demand. Following the principle that large computational problems can often be divided into smaller ones, various forms of parallel computing, from hardware dependent solutions such as multi-core and GPU programming, to distributed computing, have been proposed to provide a suitable parallel computing architecture.

Distributed computing is an important class of parallel computing, linking distant high performance computing resources through the Internet. Such systems are essentially cooperative groups of powerful computers that require both an initial investment in hardware and software as well as significant operational costs (e.g., maintenance, direct power consumption and cooling infrastructure) that are mostly energy-related. Increasing operational costs [1], combined with the need to reduce the related carbon footprint, have led researchers to explore energy-efficient alternatives for high performance computing that decrease the overall energy consumption of computation, storage, and communication. Several ideas have been explored, including PowerNap [2], which relies on the hardware ability to switch to a low power state, and GreenCloud [3], which considers migrating virtual machines between physical machines in order to reduce the total power load of a data center. However, improving energy efficiency in large scale workstations is still considered a major challenge in distributed computing [4].

Instead of using dedicated hardware for parallel computing, volunteer computing aims to use underutilized personal computational resources. Many computing devices (e.g., personal computers, tablets and mobile devices) under utilize their processing capabilities for the majority of their operational time, during which they could be used for other tasks. Recent studies show that the potential of these resources exceeds any centralized computing system [5]. Many systems have been proposed with the objective to allow volunteers to dedicate the unused computing cycles on their personal computers, such as the SETI@home project [6], JXTA [7], Xtremeweb [8], and the Berkeley Open Infrastructure for Network Computing (BOINC) [5]. BOINC has been one of the most popular volunteer computing platforms, with over 1,000,000 active computers for a large range of application areas throughout the world [9].

These solutions attempt to provide a large scale, platform-independent computing infrastructure, but most of them are limited to personal computers. However, the availability of wirelessly connected mobile devices has grown considerably within recent years, creating an enormous collective untapped computational power. The idea of integrating mobile devices into the computational grid was proposed more then a decade ago [10], when mobile computing devices such as laptops and PDAs were typically restricted by reduced processing power, memory, secondary storage, and bandwidth capabilities. The authors in [10] recognized that, even if the individual mobile devices have limited resources, considering them as an aggregated sum, they have the potential to play a vital role within distributed computing.

Nowadays, with the recent advances in the area of low powered processors, mobile devices such

as smartphones and tablets are able to perform computationally intensive operations, so that they are now considered as alternative computing platforms. For instance, a typical tablet such as the Asus Nexus 7 [11] is equipped with a 1.5 GHz quad-core CPU and 2 GB RAM which, for standard workloads, provides performance comparable to an entry-level laptop processor [12]. Although the computing capabilities of mobile processors are not as powerful as the ones of a standard desktop computer, they have been shown to be more energy efficient [13].

As a result, many traditional distributed computing platforms have attempted to extend their operation over mobile devices. For example, Hyrax [14] provides an Android application to execute jobs for Hadoop Apache on smartphones and, following the same approach, the BOINC project released an Android client [9] to include mobile devices in the volunteer computations. More recently, several distributed system architectures and frameworks, such as GEMCloud [13], CrowdLab [15] and Seattle [16], have been proposed to exploit the computational capabilities of mobile devices, while trying to address the challenges that arise from their integration into a traditional distributed architecture.

In this chapter, we present recent advances in the mobile distributed computing research field, where mobile devices are the elements that perform the computation. This computation is either assigned by a traditional remote server or a local device. In both cases, several studies show that it is feasible and beneficial in terms of both energy and execution time to allow mobile devices to participate in the distributed computation. Moreover, recent research shows promising results toward a distributed computing architecture that opportunistically harvests the computational power of volunteer mobile devices. In the remainder of this chapter, the term mobile device is used to represent a small, handheld computing device, such as a smartphone or a tablet.

The rest of this chapter is organized as follows. In Section II, we provide a classification of different parallel computing techniques, namely cluster computing, distributed computing and volunteer computing, discussing how mobile devices can provide benefit to the parallel computation. In Section III, we discuss the motivations and challenges behind the design of a mobile volunteer computing architecture, while in Section IV, we present a comprehensive review of different frameworks for parallel computing that incorporate mobile devices that are currently proposed in the literature. Section V describes the current open issues and future research directions to support mobile volunteer computing. Finally, Section VI concludes the chapter.

II. Classification of Parallel Computing

High performance parallel computing has been an approach used to increase the speed of computation by dividing the computational problem into simultaneously computable sections and processing each section on different processing units. Traditionally, these independent processing units reside on the same device (multiprocessor computing), or even on the same chip (multicore computing). On the other hand, researchers have explored new computational architectures where the processors of multiple devices are connected by a communication network and cooperate in the computational job. These architectures can be classified according to the geographical distance between the devices that perform the computation: the parallel execution of computational jobs using a group of co-located computers is typically called *cluster computing*, while the cooperation among distant computers communicating over the Internet is typically referred to as *distributed computing*. While the former relies on a reliable local area network and can be used to solve distributed computing problems that require communication among the devices executing the tasks, the latter, due to the unpredictability of the Internet, typically deals only with what are termed "embarrassingly parallel problems," where there exists no dependency (or communication) between the parallel tasks. Both of these approaches consider that the computation is distributed across dedicated devices that either require direct management or the payment of a fee for accessing the processing power. As a result, the concept of *volunteer computing* has been proposed as an alternate parallel computing system that exploits computing resources donated by general-purpose computer owners.

In what follows, we first briefly describe these three classes of parallel computing, namely cluster computing, distributed computing, and volunteer computing, and then discuss how mobile devices can provide benefit to the distributed computation.

A. Cluster Computing

Computing clusters are built linking groups of computers through a high-bandwidth low latency local area network. These computers each run their own instance of an operating system, but work together to perform a common task so that they can be viewed as a single system. The computing clusters are developed for a variety of purposes such as load balancing on web servers, computationally intensive scientific calculations, and failure safe operation on critical commercial applications.

Attached Resource Computer (ARCNET) [17] was the first commercial computing cluster, developed in the late 70s, supporting both parallel computing as well as sharing file systems.

Beowulf clusters utilize standard commodity grade computers with specialized libraries and programs that allow job sharing among them. Beowulf clusters normally run Unix like operating systems, such as BSD, Linux, or Solaris and, potentially, any PC capable of running a Unix like operating system can be used in this configuration. The cluster is organized as multiple computers serving as the worker nodes and one or more computers taking the responsibility of the server. The server controls and coordinates the computing cluster and serves as a gateway between the computing cluster and the outside world. Stone Soupercomputer[1] [18] built by Oak Ridge National Laboratory was one of the large scale successful applications of the Beowulf concept.

Due to the dependency of the physical location of the hardware, computing clusters are built to serve a limited set of users located at a particular geographical region. Hence, the demand for computational resources on these systems have a high variance due to the correlation between usage patterns. Combined with the high cost of building computing clusters, this leads to both underutilization and outage of computational resources.

B. Distributed Computing

Distributed computing overcomes the geographical limitation of cluster computing by allowing distant computers to cooperate in the execution of computational tasks. By integrating geographically diverse multiple computing clusters or individual computers, distributed computing architectures can serve a larger group of consumers with less correlated usage patterns. Although distribution and scheduling of the computing jobs across the distributed computing resources adds another layer of complexity, with the introduction of the Internet, distributed computing systems provide a fairly low cost and high performance solution to large computing problems.

Through distributed computing, computational capabilities can be offered to users as a service. In this new model of computing, also referred to as utility computing, customers can acquire large computing capabilities as needed. The computational tasks are offloaded to the service providers' computing platform, and the results are downloaded back after completion of the tasks. Many commercial instantiations of distributed computing exist today, including Amazon Elastic Compute

[1]The Stone Soup is an old folk story in which hungry strangers persuade local people of a town to give them food. It is usually told as a lesson in cooperation, especially in situations of resource scarcity.

Cloud [19]. One intrinsic drawback of this approach is that the users' performance is negatively affected by the network delay, since the entire user data and the result of the computation need to be exchanged back and forth with the distributed computing system.

More recently, a new subclass of distributed computing named cloud computing has also been proposed, and it is receiving considerable attention. Distributed computing architectures have evolved into cloud computing systems that not only undertake computational tasks but also serve as data storage systems and provide online access to computer services or resources. These resources are shared by multiple users but are usually dynamically reallocated per demand, thus maximizing the effectiveness of the shared infrastructure. Microsoft's OneDrive [20] and IBM Cloud [21] are two of the many commercial examples of this paradigm.

C. Volunteer Computing

Although distributed computing systems increase the efficiency of parallel computing, they still require a large investment for both hardware and software as well as incurring significant operational costs (i.e., maintenance, direct power consumption and cooling infrastructure). Several studies have shown that many computing devices (i.e., personal computers, tablets and mobile devices) under utilize their processing capabilities for the majority of their operational time. The potential of these resources exceeds any centralized computing system. This is the basis for volunteer computing.

The first volunteer computing project, Great Internet Marsenne Prime Search [22], was started in 1996 with the objective of using freely available software on volunteers' computers working in parallel to find prime numbers. Starting from this project, volunteer computing emerged as a result of the wide spread adoption of personal computers and the Internet. With volunteer computing, volunteers can dedicate the unused computer cycles on their personal computers to the distributed computation. This is made possible by middleware systems such as JXTA [7], XtremeWeb [8], and Berkeley Open Infrastructure for Network Computing (BOINC) [5]. BOINC was originally developed to provide support and increase security for the SETI project [6] and later extended as a platform for other distributed applications. It is now one of the most popular volunteer computing platforms with over 1,000,000 active participants [9].

D. Parallel Computing on Mobile Devices

The idea of connecting mobile devices into a parallel computing system was proposed in 2002 [10], when both their computational capabilities and diffusion where still highly limited. With the increase

in mobile device computational capabilities, different system architectures have been proposed to exploit their resources for parallel computing. The classification of parallel computing presented in this section can be extended to the case in which the mobile devices are performing the actual computations. In this regard, solutions that group nearby mobile devices using a device to device communication technology such as Bluetooth [23] and WiFi Direct [24], and distributed systems that link together distant devices through an Internet connection have both been investigated. Many traditional distributed computing architectures have recognized the widespread usage, significant computing capabilities and energy efficiency of mobile devices and have attempted to extend their operation over mobile computing platforms. For example, Hyrax [14] ports Hadoop Apache, an open-source implementation of MapReduce, to execute jobs on networked Android smartphones. A client version of BOINC was ported to an ARM/Linux platform [25] to evaluate the processing power of mobile devices, and an Android client [9] to include mobile devices in the distributed computations has also been released by the BOINC project.

III. Motivations and Challenges

There are several motivating factors that make personal mobile devices suitable for inclusion in a distributed volunteer computing architecture. However, their integration with a traditional system is not straightforward, and some important issues need to be addressed. In what follows, we first describe the motivations behind utilizing personal mobile devices as part of the distributed computing architecture, and then we present the main design challenges.

A. Motivations

The first motivating factor is the impressive rise in the number of smartphones and tablets across the world. According to a report released by the International Data Corporation (IDC) [26], in 2013 the worldwide smartphone market shipped one billion units in a single year for the first time, representing a 38.4% increase with respect to the 725.3 million units shipped in 2012. Smartphones accounted for 55.1% of all mobile phone shipments in 2013, up from the 41.7% of all mobile phone shipments in 2012. Moreover, IDC recently reported a new single quarter record of 301.3 million shipments for the second quarter of 2014 and forecasts a 23% year increase from the 1.0 billion units shipped in 2013. The progressive increase in worldwide shipments of smartphones, from the first quarter of 2011 to the second quarter of 2014, is presented in Figure 1. A similar trend has been reported for the worldwide tablet market, with a total shipment of 50 million units during the

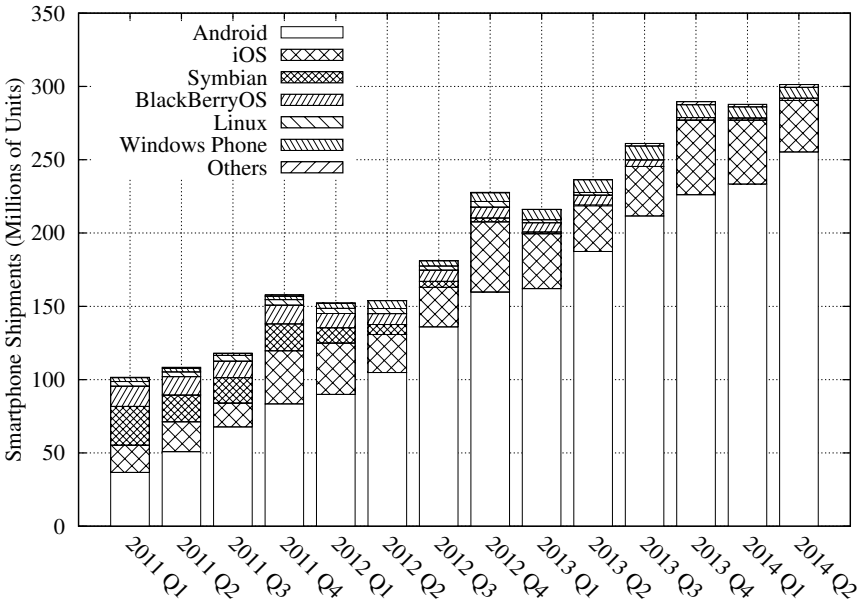

Figure 1. Quarterly worldwide shipments of smartphones and operating systems market share from the first quarter of 2011 to the second quarter of 2014. Data source: IDC worldwide quarterly mobile phone tracker [26].

second quarter of 2014, resulting in an 11% year over year increase. These trends in smartphone and tablet sales can be compared with the worldwide PC[2] shipments, that totaled 74.4 million units in the second quarter of 2014, with a year-on-year decline of -1.7% [26]. As a result, the global number of smartphones and tablets is continuously growing at a fast pace, and the global number of users outnumbers the users of conventional Laptop and Desktop PCs, reaching almost 30% of the worldwide population [27]. Thus, the computing power offered by mobile devices disseminated across the world is already substantial and is going to increase further in a dramatic fashion in the coming years.

In addition to the large amount of aggregate computing power offered by these personal mobile devices, their intrinsic energy efficient design makes them particularly suitable for the execution of computational tasks. It has been shown that a mobile processor like the Qualcomm Snapdragon S4 that powers the Nexus 7, can be over 20 times more power efficient than a commercial PC CPU such as the Intel Core i3 while, at the same time, achieving half the performance of a desktop processor in standard benchmarking tests [12]. The computational power evolution of mobile processors from 2010 to 2014 is shown in Figure 2. In fact, even if specialized high performance server architectures

[2]PCs include Desktops, Portables, Ultraslim Notebooks, Chromebooks, and Workstations.

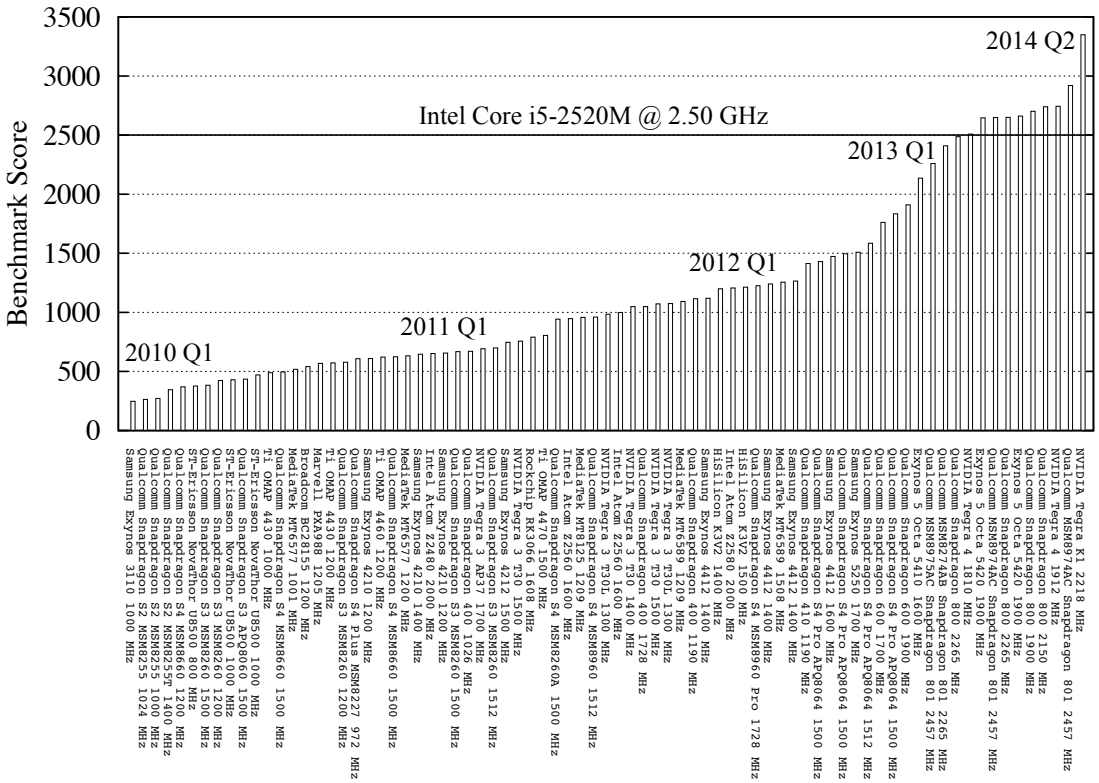

Figure 2. Benchmark scores of different mobile processors released between the first quarter of 2010 and the second quarter of 2014. Data source: Primate Labs Geekbench 3 [28].

can provide better performance per consumed power ratios, such servers consume much more power than a mobile device when they are in the idle state [13]. Moreover, a high performance server requires additional supporting infrastructure for the installation and operational expenses for related equipment (e.g., the cooling system and backup power) and maintenance, thus resulting in a higher total cost of ownership compared to mobile devices. As pointed out in [29], mobile devices already exist and thus, by using them in a distributed computing system, it is also possible to save the cost, energy and material consumption required for the production and operation of new high performance workstations. Finally, from an environmental point of view, the use of existing mobile devices should be increased. The authors in [30] estimate that only 25% of the total energy budget required for the production and operation of a mobile device is actually attributed to its usage (i.e., energy for charging the battery), while the remaining 75% is used during the manufacturing of the device.

Another motivating factor is the rising costs required for running a distributed computing in-

frastructure using workstations, which are mainly due to maintenance costs and energy charges. While the energy charges keep increasing [1], for a data-center they are based on the industrial rate, which is much higher than the consumer rate at which the mobile devices owned by consumers will operate. In addition, the authors in [31] estimate the cost for powering an Intel Core 2 Duo server to be $74.5/year, compared to $1.33/year for a smartphone with the same computational performance. Thus, using consumer smart devices for distributed computation results in significantly lower energy costs.

B. Challenges

The first problem to consider when dealing with personal mobile devices is their availability. In particular, smartphones and tablets have mobility as one of their major features, which makes them volatile computing resources. Due to mobility, the devices are not physically available in a specific location, and thus it is possible that most of them will not be available for computing purposes for extended hours. Furthermore, since these devices are battery powered, battery life is also a major concern, which can severely impact the user experience. However, in many cases, mobile devices are unused and do not move for long periods of time during a working day and, especially, at night when most users leave their devices idle while charging the battery. In the idle state, a mobile device typically runs only lightweight background jobs (e.g., email download, data synchronization and application updates) that require only minimal computation and intermittent Internet access. Thus, it is important to devise intelligent algorithms for task scheduling and distribution that take into account the device availability, relative power state (i.e., battery level and charging status) and user utilization patterns in order to maximize computing capacity without negatively impacting the standard mobile device operations.

A second challenge is the need for Internet connectivity, which is necessary for the distribution of the tasks and for the transmission of the computation results. Internet connectivity is a highly limiting factor due to both its cost and its impact on the energy consumption of the device. Moreover, even when the devices are connected to the Internet, the stability and bandwidth of the wireless network are also important elements that need to be considered. This is because channel fluctuations can result in data losses or in reductions of the useful bandwidth, and the bandwidth will then impact the total time required to distribute the computation and receive the results. While it is neither advisable nor feasible for mobile devices to employ conventional wired networking, the current wireless technologies are either limited in speed and require a substantial energy consumption

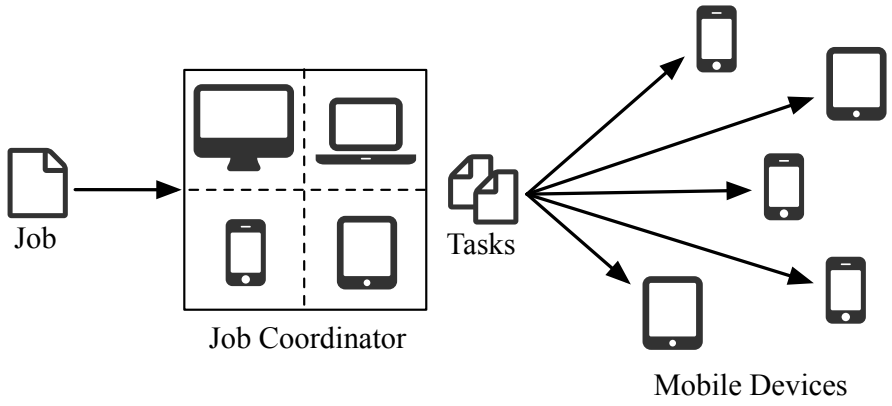

Figure 3. Example of a mobile distributed computing architecture where the job coordinator assigns tasks to the participating mobile devices.

(e.g., 3G and 4G/LTE), or are limited in range (e.g., WiFi). Thus, devising techniques to reduce the cost of communication, preserve the battery power and periodically monitor the availability and relative bandwidth of different networking technologies, is a major requirement for an efficient utilization of personal mobile devices as additional computational resources.

Another important challenge that needs to be considered when dealing with mobile devices is their heterogeneity in processing capabilities and platforms. While the latter seems to be mainly a problem of the past since, starting from 2012, more than 90% of the mobile device OS market is dominated by Android and iOS [26] (during the second quarter of 2014, 84.7% Android and 11.7% iOS [26]), the processing capability varies greatly among devices, especially for the Android market. This makes the design of a general architecture that can leverage advantages of a vast majority of software/hardware platforms very difficult. As an example, when assigning tasks to mobile devices, the computing architecture needs to take into account the hardware resources available at the device and assign computationally intensive tasks to more powerful devices.

IV. Mobile Distributed Computing Architectures

Modern mobile devices, such as smartphones and tablets, have become powerful and energy efficient computing architectures, that are both widely available and underutilized for long periods of time. As a result, multiple approaches for integrating mobile devices into a parallel computing infrastructure have been proposed and are currently receiving considerable attention. These studies can be broadly classified in two main categories, depending on the particular entity that is responsible

for the management of tasks that need to be executed by the participating devices. This element can either be a remote specialized server that communicates with the mobile devices through an Internet connection, or it can be a mobile device itself, that exploits the presence of other geographically close devices to solve computationally intensive tasks. According to this division, we refer to the first scenario as "Server Driven Mobile Distributed Computing," while we call the latter scenario "User Driven Mobile Distributed Computing." We note that the server driven approach is, in fact, an extension of a traditional distributed computing architecture, where mobile devices can participate in the distributed computation alongside standard PCs. The user driven case, instead, represents a viable way to perform intensive computing when an Internet connection is not available or it is undesirable because of communication delay. For example, many applications in the area of tactical military communications, search and rescue operations, and sensor network operations require computing intensive algorithms, such as image and signal processing, at remote or isolated locations that frequently neither have direct access to the Internet nor are in the vicinity of other devices with Internet access. In both cases, the device that "drives" the computation is considered to be the job coordinator, since it is in charge of the task distribution process and is responsible for the reception and organization of the results of the tasks' execution. An example of such a mobile computing architecture is presented in Figure 3.

In what follows, we provide a literature review of the different frameworks that have been proposed to use mobile devices for executing tasks in distributed computing. All of these systems follow a similar architecture and communication protocol: a mobile device is connected through a suitable radio communication technology to the job coordinator, it receives the tasks to be computed and, after the execution is completed, the mobile device sends the result of the computation back to the job coordinator. It is important to note that the idea of enabling the execution of rich applications on mobile devices by offloading the computation (or part of it) and storage to a distributed architecture composed of traditional computers has also been proposed. This type of service is typically referred to as mobile cloud computing [32].

A. Server Driven Mobile Distributed Computing

1) Mobile OGSI.NET [33]: Mobile OSGI.NET was one of the first attempts to connect mobile devices to a distributed computing architecture. The goal of this framework is to provide a way for mobile and non-mobile devices to collaborate in the execution of resource-demanding applications. The OGSI architecture consists of three components: the Mobile Web Server that handles the

message exchanges between the device and the remote server, the Grid Services Module that implements the core processing necessary to execute applications, and the Grid Services, that represent the particular applications that will be executed by the device. In addition, the Grid Services Module monitors the resources of the mobile devices (like, e.g., the battery level) and decides if the mobile device is able to perform the computation, or if it is better to pass the task to another device. Experimental results show that the total time required to compute a set of tasks decreases as the number of devices increases. Moreover, they show that the energy consumption can be efficiently distributed between the devices.

2) Hyrax [14]: Hyrax is a platform derived from Hadoop Apache, an open source implementation of MapReduce[3], that supports distributed computing on Android devices. The basic idea behind Hyrax is to allow a heterogeneous network of smartphones and servers to cooperate in the execution of computing jobs. The framework has been designed to provide an abstraction of the available resources, thus being able to scale with the number of devices and tolerating node connection and departure. The performance of Hyrax in terms of both execution times and resource usage was evaluated with a testbed of 12 Android smartphones. Although the performance of Hyrax is poor for CPU-intensive tasks, it demonstrates the feasibility and scalability of the proposed framework. In addition, the advantages of using Hyrax as an infrastructure for applications that use mobile data have been investigated through the implementation of a distributed multimedia search and sharing application. The authors of [14] stated that Hyrax had not been optimized for battery efficiency, but in several tests it was shown to use significantly less power than a video recording and downloading application.

3) Computing While Charging [31]: Computing While Charging (CWC) describes and evaluates a scenario in which a company uses the mobile devices provided to its employees for the execution of parallelizable tasks. The main idea behind CWC is that using mobile devices for work-related computing can potentially reduce not only the capital investment in servers but also the cost of energy, since a smartphone can be up to 20x more efficient than a standard server. Thus, the authors of [31] propose a framework where the phones are used for the computation only while being charged, so that the user is not disturbed by the computations. Moreover, while charging the phone has a high probability of being connected to the Internet through a WiFi Access Point. The application monitors the user interactions with the phone and, if a user uses the phone while

[3]MapReduce is a programming framework for data-intensive cloud computing on commodity clusters developed by Google.

it is computing a task, the task is interrupted and migrated to a different phone so that the task computation does not have any impact on the user. Moreover, the application incorporates an algorithm that monitors the charging patterns: a test performed on 15 volunteers showed that the users charge their phones predominantly during the night and for an uninterrupted period of several hours. They also ran other experiments to evaluate the impact of the network connectivity on the task completion time, showing that simply accounting for the CPU clock speed results in poor task completion times. CWC also includes an algorithm to predict the time it takes for the tasks to be completed and three task distribution methods with different complexity. The main experiment involved 18 Android phones with different CPU clock speeds and different network connectivity technologies, and it showed that a greedy scheduler is approximately 1.6 times faster than the other tested schedulers.

4) jUniGrid [34]: jUniGrid is a lightweight framework that allows the integration of mobile devices into heterogeneous desktop distributed computing systems to solve high complexity computational problems. jUniGrid introduces two separate applications, that correspond to two functional roles: the Task-Submitter (TS) and the Node-Application (NA). The node application is installed on the devices that execute the tasks, while the task submitter runs on the device that creates the tasks, stores them in a task queue and assigns them to the nodes according to a First In First Out policy. Moreover, jUniGrid works based on a split/merge algorithm. It provides the user with the flexibility to split the job and merge the results according to the requirement of the particular job. This split and merge is accomplished by TS, that also implements all the functionalities for job allocation, monitoring and result aggregation. Thus, the TS is installed on the device that creates task queues and sends them to the nodes in a FIFO fashion, where the first device to be connected receives the first task output, while the node application is installed on the devices that execute the tasks. The paper shows the gains in term of job execution times that can be achieved by allowing mobile devices to participate in the distributed computation. However, the focus of the paper is to provide a basic generalized grid mechanism for cooperative multi-platform processing and does not provide any detail about the specific implementation.

5) Ocelot [35]: Ocelot is a distributed mobile computing platform that leverages mobile devices to execute lightweight computational tasks generated from a Wireless Sensor Network (WSN). Ocelot is modeled after the Berkley Open Infrastructure for Network Computing (BOINC) with the exception of employing smartphones and tablets rather than workstations and server machines, as they reduce the maintenance costs and the power usage. For testing purposes, Ocelot was integrated

with a WSN that is deployed to monitor indoor environmental conditions in a building. More specifically, Ocelot was used to monitor and analyze the electrical power consumption and the environmental emissions within the building. In this setting, Ocelot's clients (the mobile devices) are attached to sensors that exchange data through WiFi Direct and/or Bluetooth. Although the clients themselves cannot gather sensory data, they can partition and efficiently process the data through parallel computing. Ocelot's clients serve as nodes that request and receive tasks from a server through XML files. One of Ocelot's main features is having multiple servers to insure an efficient task distribution, as one of the servers is consistently storing the battery status of the nodes to make sure that the scheduling server sends tasks to only those nodes with sufficient power. Once a node receives a task, it will execute the code, which is usually written in Java or C. Currently, Ocelot allows the mobile devices to act only as clients that execute tasks, but allowing the mobile devices to also become servers for the task distribution is considered as future work.

To prove its advantage in reducing energy consumption, Ocelot has been compared against traditional computers. Results show that, while laptops and desktops were 5 times faster than mobile devices, mobile devices consume up to 86% less energy. In addition, adding more devices to the client pool dramatically lowers the total task completion time. As a result, Ocelot proves that it is feasible to distribute tasks among mobile devices and provides considerable energy and cost savings with respect to a system that uses standard computers.

6) CANDIS [29]: CANDIS is a framework that distributes computing tasks to mobile devices as well as normal desktop or server hardware and provides an efficient method of reducing costs by taking advantage of the fluctuating energy market prices. The authors in [29] recognized that distributing computing tasks to mobile devices has been extensively studied. Thus, they focus on devising a computational infrastructure that is able to further reduce the computation costs and energy consumption. CANDIS is a Java-based framework and is thus able to run existing Java code, and desktop, server and Android mobile devices that support Java can be easily connected in the cloud. In such a hybrid cloud, the server compiles the tasks and constructs a scheduler to allocate and distribute the tasks. The paper shows that, while equally dividing the tasks among the available devices might be easier to implement, distributing much smaller tasks results in faster execution times but increases the communication overhead. Thus, CANDIS implements a more efficient allocation method where the server, before assigning the actual tasks, estimates the capability of each client by assigning a benchmarking task and using the results to improve the task allocation scheme. In addition, CANDIS uses information about the price of electricity, which

fluctuates dramatically throughout the day (on average, prices are usually much lower late at night and early in the morning). The authors in [29] conclude that using CANDIS on a large scale not only allow to save money but can also stabilize the electric energy consumptions.

7) ANGELS [36]: ANGELS is a framework that allows mobile devices and computers to cooperatively participate in the computation of analytical data. This framework allows the parallel execution of jobs on a set of nodes that can be either mobile devices or standard PCs. For the evaluation of the framework, a "text search" application, in which mobile devices and servers had to find a specific word in a large text file and an algorithm to estimate the value of π have been considered. Experimental results show that the tasks' latency can be substantially reduced when the tasks are distributed among the mobile devices. The framework does not consider the impact of the computation on the user experience. Moreover, the task distribution process considers a simple distribution scheme, in which tasks are assigned as soon as a device becomes available.

B. User Driven Mobile Distributed Computing

1) Serendipity [37]: Serendipity enables a mobile computation initiator to use the computational resources of nearby mobile devices to speed up the computation and preserve some energy. Serendipity improves the mobile device's computational experience by applying optimizing algorithms that minimize local power consumption and/or decrease the computation completion time, while taking into account the constraints of the intermittent communication links such as limited contact duration, limited transfer bandwidth, and completion-time unpredictability. Serendipity follows what is called a "PNP block paradigm": the job is pre-processed and divided into n parallel task programs, and the results of the execution of the tasks are finally merged by a post-process algorithm. The goal of this design is to have an initiator disseminate a task (pre-process) to the computational nodes (task programs) it encounters based on the estimated completion time or energy consumption, and finally coalesce the data it receives (post-process).

To do so, three algorithms have been presented: 1) a WaterFilling method where the initiator knows when to contact the nodes and has access to the nodes' profiles in order to predict the number of tasks a node can execute and the time required to process them; 2) a Computing on Dissemination (COD) method, where the initiator does not know the contact time but has access to the nodes' profiles; and 3) the Unpredictable Computing on Dissemination (upCOD), where both the contact time and the nodes' profiles are unknown.

Name	Year	Contributions	Task Distribution	Operating System	Applications
Mobile OSGI.NET	2004	Porting of OGSI.NET to mobile devices	Homogeneous tasks. FIFO queue	Microsoft PocketPC 2003	Prime numbers search
Hyrax	2009	Porting of Hadoop to Android devices	Homogeneous tasks. FIFO queue	Android	Distributed multimedia search; Content sharing
Computing While Charging	2012	Profiling charging behaviors, scheduling algorithm, migration of tasks across phones	Heterogeneous tasks. Greedy algorithm based on the Minimum Makespan Scheduling problem with task migration.	Android	Prime numbers search; Word searching; Photo pixels blurring
JUniGrid	2013	Generic framework API for developing grid applications	Homogeneous tasks. FIFO queue	JAVA	DNA sequence matching
Ocelot	2013	Distributed computing system that uses mobile devices as computing resources.	Homogeneous tasks. FIFO queue.	Android, iOS	Dynamic life cycle assessment of a building
CANDIS	2013	Distributed computing system that uses mobile devices and traditional computer as computing resources.	Heterogeneous tasks. Scheduler based on the device computational capability and the energy market prices.	Android	Distributed brute force hash-cracking; XML to JSON conversion
ANGELS	2014	Framework that allows the remote execution of programs within mobile devices. The focus is on the processing of IoT analytical data.	Heterogeneous tasks. Tasks are assigned according to the device computational capability.	Android	Text search; π value estimation

Table I

SERVER DRIVEN MOBILE DISTRIBUTED COMPUTING IMPLEMENTATIONS.

Serendipity has been implemented on Android and showed substantial performance gains when compared to executing tasks locally on the initiator's mobile device. While in all the experiments, the WaterFilling method performed better than COD and upCOD, experimental results show the clear benefit of disseminating tasks on Serendipity rather than executing them locally, especially when the number of tasks exceeds 100. Moreover, Serendipity was able to speed up computation up to 3 times compared to local conventional computing. In addition, Serendipity increases the battery life of mobile devices and allows the saving of a significant amount of energy by distributing the computation between different devices.

2) Honeybee [38]: Honeybee is a framework that deals with both human and machine computation, where human computation represents a set of operations that require human interaction, like filling out a personal survey form, while machine computation is a generic computer algorithm, like word searching or number sorting. The Honeybee framework distributes a computational intensive task, like a face detection algorithm, among several mobile devices. Honeybee's focus is on keeping the smartphone busy, in the sense that, as soon as one computation is completed, the device is allowed to steal tasks from another slower device. The proposed implementation also focuses more on getting as many tasks done as possible and does not provide a customizable, user friendly interface. Moreover, to each task is assigned a deadline that the mobile device has to satisfy in order to continue getting tasks. If the deadline is not met, the task is passed to another device. Honeybee has been implemented on Android, using Bluetooth as a local communication technology. Experimental results show that managing local connections severely impacts the delegator throughput. As future work, the authors are planning to support different types of D2D communication technologies, e.g., WiFi Direct.

3) Unity [39]: Unity represents a system architecture that allows a group of mobile devices to share the workload required to download a data file from the Internet. With this approach, each device downloads small parts of the file and then shares those parts with the other members of the group so that every device will eventually get the complete content. Leveraging short-range technologies such as WiFi and Bluetooth, Unity allows a coordinator to communicate with its peers to split the download as well as restarting it from the point where it stopped in case of a failure.

Unity has been implemented on Android smartphones that have either WiFi or Bluetooth capabilities. Unity employs WiFi HotSpot, an Android utility that uses 802.11 infrastructure mode to allow the coordinator to act as a WiFi AP and all other peers to be connected as clients. As a result, the WiFi HotSpot functionality allows the coordinator to stay awake for the entire duration,

while peers are in power saving mode, consuming a negligible amount of energy. After an initial connection phase, where the devices connect to the coordinator, the coordinator determines the size of the file through an HTTP request, divides the load, and then sends a control message containing the file URL to its peers. Subsequently, each peer starts to download its share of the file using its own data connection and sends the data blocks to the coordinator, which collects all the blocks, reconstructs the entire file and distributes it to all the peers. Unity also includes a task distribution scheme that takes into account the variability of the peers' cellular network conditions for better distributing the workload between the mobile devices. Experimental results show an improvement in download speeds up to 27%. In addition, a variations of Unity called Unity-Cloud has also been presented. In Unity-Cloud, a remote server coordinates the formation of the peer to peer group and assigns to each device the part to be downloaded according the relative cellular conditions. As soon as the peers are geographically close, the cloud coordinates the local blocks sharing for reconstructing the original file.

4) DRAP [40]: DRAP proposes a mechanism to group volunteer mobile devices into high performance decentralized computing systems. The idea behind this work is to create an infrastructure where mobile devices in close geographical proximity can form a cloudlet, and share resources with each other and with other nearby devices. The concept of a cloudlet has been introduced in [41], and represents a trusted, resource-rich computer or cluster of computers that is well-connected to the Internet and available for use by nearby mobile devices. In DRAP, the cloudlet is represented by a cluster of mobile devices that provides storage capability and computational resources to the other devices in the network. The framework monitors the movement of the participating nodes and implements all the functionalities required to connect the nodes in the network and enable the communications. For routing the communications between the devices, DRAP uses a modified version of the Ad Hoc Distance Vector (AODV) routing protocol. DRAP also includes an algorithm that uses the mobile devices' resources to determine the subset of nodes that should be selected to become part of the cloudlet. Computer simulations prove the feasibility and performance gain of the proposed architecture, for different types of applications. The implementation of DRAP in real life mobile devices is considered as future work.

C. Mobile Volunteer Computing

The architectures presented in the previous sections can all be adapted to allow volunteer users to participate in the parallel computation. However, architectures explicitly designed with the objective

Name	Year	Contributions	Task Distribution	Operating System	Applications
Serendipity	2012	Distributed computing system that exploits nearby mobile devices	Heterogeneous tasks. Three schemes with different complexity.	Simulations on Emulab	Speech-to-text application
Honeybee	2013	Framework API to support job sharing and crowd-sourcing among mobile devices. Work stealing to achieve load balancing	Heterogeneous tasks. Tasks assigned at random. Scheduler that attempts to minimize the idle time	Android	Face detection; Mandelbrot set generation, Collaborative photography
Unity	2013	System architecture that enables collaborative downloading across co-located mobile devices.	Heterogeneous tasks. Tasks (data to be downloaded) are assigned according to the cellular network conditions	Android	Collaborative file download
DRAP	2014	Cluster formation of volunteer mobile devices for distributed computation. The focus is on how to best group the devices based on their capabilities.	High level description of a Cloudlet Manager that handles task distribution. The details about the task distribution process are not provided	Simulations on ns-3	Testing of the cloudlet formation algorithm

Table II

USER DRIVEN MOBILE VOLUNTEER COMPUTING IMPLEMENTATIONS.

of realizing a volunteer computing system by interconnecting personal mobile devices through the Internet have also been developed.

1) BOINC on Mobile Devices: The first attempt to extend the participation in volunteer computing to mobile devices dates back to 2007 [25], with researchers working on getting the application SETI@home [6] (and other scientific programs) to run efficiently on ARM processors [42]. Starting from this feasibility study, different Android applications [9], [43]–[45] and a prototype implementation for iOS [46] have been proposed to extend the participation in BOINC projects to mobile devices. In February 2014, the integration of Android devices into the BOINC system has been extensively promoted with the campaign HTC Power to Give [47], an initiative that aims to create

a supercomputer by harnessing the collective processing power of Android smartphones. As of September 2014, the HTC Power to Give application has been installed on 1.7 million devices [48].

2) CrowdLab [15]: The idea of creating testbeds by interconnecting volunteer mobile devices has also received considerable attention [15], [49], [50]. In particular, CrowdLab [15] is a testbed architecture that utilizes volunteer mobile resources to offer features common to infrastructure-based testbeds. CrowdLab allows the execution of guest code on participating mobile devices through hardware virtualization, it supports low-level access to the radio device and the concurrent scheduling of co-located applications. Volunteers contribute resources to CrowdLab in the same way that users contribute spare resources to BOINC. The CrowdLab architecture uses a centralized remote server for tracking the experiments and the current available volunteer resource contributors, and a decentralized local task coordinator that is responsible for the scheduling and task distribution to nearby devices. CrowdLab includes an algorithm to limit the amount of energy that each application can consume in a certain time period. According to this scheduling scheme, a device will not participate in the distributed computation if the owner is actively using it, and it allows the user to set a daily resource budget for running experiments as a percent of battery capacity or as a period of participation.

3) Seattle [16]: Seattle [16] has been proposed as a distributed computing platform that exploits heterogeneous volunteer devices for educational and research purposes, that supports different operating systems and architectures. Seattle is a general purpose learning platform based on the Python programming language that allows users to develop and test different types of applications ranging from networking to cloud computing. The objective of this platform is to provide researchers and educators the ability to create application prototypes and to evaluate their performance on a wide range of devices distributed around the world. Seattle follows an open source philosophy, and it embraces the heterogeneity of today's end user environment, thus providing a unique environment that is not available on other testbeds. A recent study showed that more than 20,000 devices are currently contributing their resources to Seattle, with more than 500 being mobile devices [51].

4) GEMCloud [13]: More recently, GEMCloud (Green Energy Mobile Cloud) [13] has been proposed as a distributed system that utilizes energy efficient personal mobile devices as computing resources instead of desktop computers. Mobile devices are considered to be particularly appealing because of their increasing computing capabilities, great popularity and diffusion as well as for the fact that they can potentially provide energy savings with respect to standard computers. The vision of GEMCloud is to adapt the traditional distributed computing infrastructure by shifting the

load of the computation to mobile devices. GEMCloud follows a traditional volunteer computing architecture, where a remote server distributes the tasks to participating devices through an Internet connection. The task distribution is based on the device characteristics and customizable user preferences. In GEMCloud, the user experience is particularly important, and the user is allowed to finely control how much and when to contribute to the distributed computation, including settings on battery level, charging vs. not charging, WiFi vs. 3G/4G communication, and device temperature. Experimental results show a comparison between the completion time and relative energy consumption of different types of tasks and different computing devices. While the mobile devices are always slower than a high performance workstation, GEMCloud shows that some mobile devices have performance comparable to a standard computer, while always consuming much less energy. The GEMCloud application is available for download in the Google Play store.

Starting from the GEMCloud implementation, the authors in [52] presented a computational infrastructure that extends the ability of mobile devices to participate in volunteer computing through ad hoc networking. The architecture presented in [52] overcomes the intrinsic requirement of Internet connectivity to participate in volunteer computing by introducing decentralized job coordinators. These job coordinators, referred to as task distribution points, are mobile devices directly connected to the Internet that are able to invite other devices to join the computation via device to device communication. Experimental results show that allowing for additional devices without Internet connectivity to participate in the computation reduces significantly the overall time required for the execution of the tasks, with only minor additional energy consumption at the decentralized job coordinators.

V. Open Issues and Future Research Directions

In Section IV we presented the current state of the art for distributed mobile volunteer computing research. While the different implementations have shown the feasibility and highlighted the substantial performance gains that can be achieved by exploiting personal mobile devices as additional computational elements, there are still several issues that must be considered. The main challenges arise from the intrinsic heterogeneity of resources involved in a distributed mobile system. Addressing these challenges requires the design of an efficient and reliable middleware, tailored to the requirements and challenges of a heterogeneous system. Moreover, for a successful diffusion of volunteer mobile computing, it is necessary to promote the users' participation and to reassure the users that the application will not harm their devices nor compromise their privacy or

Name	Year	Contributions	Task Distribution	Operating System	Applications
Seattle	2009	Distributed computing and general purpose learning platform	Application specific tasks	Maemo Linux	Multiple applications
BOINC	2011	Porting of the BOINC client to Android devices	Project specific tasks	Android	Multiple scientific research projects
CrowdLab	2011	Testbed architecture based on volunteer mobile resources	Application specific tasks	Android	Multiple applications
GEMCloud	2013	Distributed computing system that uses mobile devices as computing resources. Evaluation of computing power and energy efficiency of mobile devices	Heterogeneous tasks assigned at random. FIFO queue subject to user preferences.	Android	Protein structure predictions

Table III

MOBILE VOLUNTEER COMPUTING IMPLEMENTATIONS.

the performance of their device with regard to its primary purpose. In what follows, we describe some open issues that require additional research in order to improve the overall experience of the different elements of a distributed mobile volunteer computing system.

A. Incentive Model

Using mobile devices can provide a substantial gain in terms of both computational power and energy consumption. However, a mobile volunteer computing architecture is based on the assumption that the devices are distributed among different users that donate their spare cycles. While there are philanthropic users that are willing to participate in research projects for major causes in the fields of medicine, science and the environment, it is expected that many participants would not allow the platform to utilize their personal devices without any limitation for free for an extended period of time. Therefore, it is important to consider some form of compensation or economic incentives that will motivate the user to participate in the distributed computation.

Traditional methods for grading and compensating users for their collaboration are based on the absolute computational power of the device, thus preferring devices with high resources over users

with limited hardware. Effort-based mechanisms for resource sharing in collaborative applications have also been proposed, and have been shown to increase fairness in heterogeneous systems [53], [54]. An analysis of the strategies to pursue in order to improve volunteer participation and retention rates has been presented in [55]. In particular, based on a qualitative study of users in a volunteer computing project, the authors in [55] argued that a volunteer architecture should include a scoring system that is easy to understand by the users and provides an accurate and reliable indication of the user's contribution to the project. Moreover, regular feedback about the distributed computation progress should be made publicly available to reassure the volunteers that the computation in which they are involved is worthwhile and is producing certified scientific results. As an example, positive results for incentives in the form of social contracts for common goals and immediate benefits have been presented in [39] and [56].

In order to further enhance participation in the volunteer computing system, researchers have also moved from social mechanisms to monetary based schemes. In this regard, pricing has shown to be effective in encouraging sharing of computing devices, and several solutions have been proposed, such as Flat Rate [57], Auction-based [58], Stock-Market and Micropayment approaches [59]. The integration of distributed credit-based schemes with the task scheduling policy has also recently been investigated [60]. Positive indications on the usefulness of monetary incentives have been provided in [61].

Several works have been proposed for increasing the participation of users in a traditional volunteer computing system. However, when considering mobile devices, the users not only will incur some cost during their contribution to the distributed computation, such as battery energy and data transmission costs, but their normal use of the mobile devices may be affected when processing tasks. Thus, it is important to revise the existing methods to evaluate the impact of the participation in the distributed computation on the user experience and remunerate the users accordingly. These incentives could form the foundation of a new business model and enhance the efficient use of computational devices.

B. Device Heterogeneity and Task Distribution

When dealing with devices with different capabilities in terms of power, memory and processing capabilities, it is particularly important to devise algorithms that can explore the available resources and maximize the results that can be achieved through the distributed computation. As different mobile devices have different capabilities and follow different user usage patterns, it is very difficult

to group these devices according to their potential contributions to the distributed architecture and, at the same time, devise a task distribution method that is able to classify the mobile devices and distribute the computational workload accordingly.

Moreover, the way in which mobile devices connect to the volunteer distributed architecture creates additional challenges, starting from the availability of an Internet connection itself. Even when available, the heterogeneity of the network services and communication technologies reflect great variations in communication speeds and latencies that negatively affect the performance of both the mobile devices and the distributed computation infrastructure. This is especially problematic when dealing with time sensitive applications that require devices that are not only able to perform the assigned computation in a short amount of time, but that can capitalize on a fast and reliable connection to receive the task and reply with the results. As a result, scheduling tasks on an unpredictable network requires complex rules, which makes it difficult to estimate the availability and the response and transfer time of the mobile devices.

Given the above, identifying and classifying the total available resources are required for devising an efficient task distribution process. As described in [62], the device attributes can be divided into two categories, namely static and dynamic. The static attributes reflect the intrinsic characteristics of the device and will not change over time like, for example, CPU frequency, number of cores and memory. Dynamic attributes, instead, exhibit dynamic behavior and change over time like, for example, available CPU or number of cores, network connectivity and battery level. We note that the user usage patterns can be seen as a particular dynamic attribute that affects multiple device specific resources. Clearly, static attributes are easy to retrieve because they do not change over time. However, tracking the dynamic changes in resource availability requires a suitable protocol that is able to gather this information with a certain accuracy and, most importantly, limit the impact on the user experience. For an overview of existing resource discovery algorithms for distributed computing systems, we refer the reader to [63] and references therein.

The task distribution policy uses the collected and analyzed mobile device attributes, as well as the user usage patterns and preferences, to efficiently manage all the available resources. This process represents the core of the entire volunteer mobile computing infrastructure. This is because the task distribution dictates the performance of the entire system and can, in fact, determine the success or failure of the adoption of mobile devices as computational devices. The task distribution process is thus fundamental in determining how to divide the job into computational tasks and in finding the best criteria for assigning these tasks to the different clients. Different solutions for a

traditional distributed computing system have been proposed, showing promising results in real life testing. An extensive discussion of the existing task distribution policies can be found in [62].

While all the existing schemes for resource discovery and task distribution can be extended to support the presence of mobile devices, they have been designed focusing on standard computers. Thus, as described earlier, the different characteristics of mobile devices when compared to traditional computing devices can potentially require a substantial revision of the existing policies. Finally, the need for more intelligent algorithms that are able to divide and distribute tasks with different requirements, to predict the task execution times when using devices with different capabilities, to distribute the load accordingly, and to adapt the start and stop of the computation according to the user usage or the resource availability is considered an open problem even for a traditional volunteer computing system [62].

C. Result Validation and Aggregation

A volunteer computing system relies on a divide and conquer approach in which the job is divided into several computational tasks that can be processed in parallel, and the results of the distributed computations are then merged to solve the initial problem. These types of algorithms are able to determine the quality and validity of the results only a posteriori, most of the time even after the actual merging of the results of the different tasks, potentially causing the loss of a large amount of computation. Moreover, in a scenario with heterogeneous devices that are potentially executing tasks with different complexity and requirements, it is even more difficult to aggregate and validate the results. Thus, when dividing and assigning the tasks it is important to design a mechanism to verify and, if necessary, discard erroneous intermediate partial results so that the entire computation is not wasted.

Another limitation of the existing systems is that the mobile devices submit their results only at the end of the computation. Most of the implementations presented in Section IV consider the user experience as the most important design constraint, and propose different techniques to stop or migrate the computation to other devices to limit the impact of the task execution on the standard mobile device operations. In situations where the computation is terminated or the migration fails, the partial computation done by the devices is lost. Thus, a framework that exploits partially completed tasks would greatly improve the overall system performance.

Most of the existing volunteer computing systems, as well as the architectures described in Section IV, consider a centralized system, with communication going through a single server

or coordinator that fulfills the role of job scheduler and handles the task distribution and result validation. As a result, these systems either do not implement any result aggregation and validation mechanisms or create a considerable overhead on the server and are thus limited to embarrassingly parallel applications [64]. Recent solutions tolerate clients' failures by assigning the same task to multiple devices and by replicating the intermediate and final output of the computations across different devices [64], [65]. Moreover, the validation of the task results is performed at the server through replication and majority voting. According to this approach, each task has at least two replicas, running on different devices and, upon reception of sufficient results for the same task, the server is able to consider it valid if a strict majority of clients return the same output [64].

D. Security, Privacy and Data Confidentiality

Security is an essential factor for the success of the adoption of personal mobile devices as computing elements in a distributed infrastructure. As a matter of fact, creating a secure and trustable environment is an open issue even in traditional volunteer computing systems [62], both for the tasks coordinator and for the devices that perform the computation. On one hand, the clients need to trust the middleware of the distributed computing framework that it will not harm their devices or compromise their privacy and that it is honest in the description of the credit, processing and storage resources associated with the computation. Moreover, the task distributor needs to implement appropriate security policies so that, for example, hackers may not gain access to the resources of the mobile devices and use them for malicious activities. On the other hand, the task distributor needs to implement some trusting policies in order to discriminate malicious users that are returning false results, as well as mechanisms that prevent the users from gaining access to the content of the assigned tasks and related data. Moreover, dealing with all these problems is complicated by the fact that the devices involved in a distributed system communicate through the Internet, which is intrinsically insecure.

Several security mechanisms have been proposed in the literature in order to reduce the vulnerability of distributed computing systems to malicious attacks. In particular, result validation techniques based on voting [5], credibility [66] and spot-checking [67] have been proposed and are now included in many architectures. In order to prevent malicious code distribution, many systems integrate code signing techniques. As an example, each BOINC project periodically changes its code signing key, and uses the old private key to generate a signature for the new public key [9]. Techniques for preventing the intentional or accidental abuse of participating hosts by the volunteer

architectures include account based sandboxing (that is, the middleware runs under an unprivileged account and it is not able to access or modify any other data on the computer) and middleware resources monitoring to detect if the resource usage is not in accordance with the application specification [9], [68]. Most of the available systems still lack solutions to protect against users accessing the content of the assigned tasks and related data, that both typically reside in cleartext in memory [9], [62]. An extensive presentation of security threats for a distributed computing system has been recently presented in [69] and [70].

While different security methods have been proposed for a general volunteer computing system, security is still considered an open issue [9], [62], [68]. Traditional cluster and distributed computing architectures, instead, represent more controlled systems and implement complex security mechanisms [68]. The possibility of porting some of these methods to volunteer computing systems and, consequently, the development of lightweight mechanisms for securing the communications and to guarantee data integrity over a heterogeneous volunteer system of mobile devices is still under investigation.

VI. Conclusions

In this chapter, we presented recent advances in the mobile volunteer computing research field, where mobile devices are the elements that perform the computation. We described the motivations behind the adoption of personal mobile devices as computing resources and the challenges that come from their integration in a distributed system. In addition, we discussed some open issues that arise in the integration of heterogeneous mobile devices into a distributed computing infrastructure, as well as security issues and possible improvements to the task distribution process and the result validation and aggregation.

This chapter provides the research community with a comparison of the existing architectures and a description of the current open problems that will help in the design of future distributed mobile volunteer computing systems.

References

[1] "U.S. Energy Information Administration, Short-term energy outlook." [Online]. Available: http://www.eia.gov/forecasts/steo/pdf/steo_full.pdf

[2] D. Meisner, B. T. Gold, and T. F. Wenisch, "Powernap: Eliminating server idle power," in *Proc. of ACM ASPLOS*, Washington DC, USA, Mar. 2009.

[3] L. Liu, H. Wang, X. Liu, X. Jin, W. He, Q. Wang, and Y. Chen, "Greencloud: A new architecture for green data center," in *Proc. of ICAC-INDST*, Barcelona, Spain, Jun. 2009.

[4] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *Journal of Internet Services and Applications*, vol. 1, no. 1, pp. 7–18, May 2010.

[5] D. P. Anderson, "BOINC: A system for public-resource computing and storage," in *Proc. of IEEE/ACM GRID*, Pittsburgh, PA, Nov. 2004.

[6] D. P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer, "SETI@home: An experiment in public-resource computing," *Commun. ACM*, vol. 45, no. 11, pp. 56–61, Nov. 2002.

[7] L. Gong, "JXTA: a network programming environment," *IEEE Internet Computing*, vol. 5, no. 3, pp. 88–95, 2001.

[8] G. Fedak, C. Germain, V. Neri, and F. Cappello, "XtremWeb: a generic global computing system," in *Proc. of IEEE/ACM CCGrid*, Brisbane, Qld., May 2001.

[9] "BOINC," BOINC's Homepage, Last time accessed: October 2014. [Online]. Available: http://boinc.berkeley.edu/

[10] T. Phan, L. Huang, and C. Dulan, "Challenge: Integrating mobile wireless devices into the computational grid," in *Proc. of ACM MobiCom*, Atlanta, Georgia, USA, Sep. 2002.

[11] "Asus Nexus 7 (2013)." [Online]. Available: http://www.asus.com/Tablets_Mobile/Nexus_7_2013/

[12] "CPUBoss - Compare CPU to see which is faster." [Online]. Available: http://cpuboss.com

[13] H. Ba, W. Heinzelman, C.-A. Janssen, and J. Shi, "Mobile computing - A green computing resource," in *Proc. of IEEE WCNC*, Shanghai, China, Apr. 2013.

[14] E. E. Marinelli, "Hyrax: Cloud computing on mobile devices using mapreduce," Sep. 2009.

[15] E. Cuervo, P. Gilbert, B. Wu, and L. Cox, "CrowdLab: An architecture for volunteer mobile testbeds," in *Proc. of COMSNETS*, Bangalore, India, Jan. 2011.

[16] J. Cappos, I. Beschastnikh, A. Krishnamurthy, and T. Anderson, "Seattle: A platform for educational cloud computing," in *Proc. of ACM SIGCSE*, Chattanooga, TN, USA, Mar. 2009.

[17] "Attached Resource Computer (ARCNET)." [Online]. Available: http://www.arcnet.com/

[18] "The Stone SouperComputer." [Online]. Available: http://www.extremelinux.info/stonesoup/

[19] "Amazon Elastic Compute Cloud (Amazon EC2)." [Online]. Available: http://aws.amazon.com/ec2/

[20] "Microsoft OneDrive." [Online]. Available: https://onedrive.live.com/

[21] "IBM Cloud." [Online]. Available: http://www.ibm.com/cloud-computing/us/en/

[22] "Great Internet Mersenne Prime Search." [Online]. Available: http://www.mersenne.org

[23] Bluetooth Group, "Specification of the Bluetooth system," Jun. 2010.

[24] Wi-Fi Alliance, P2P Task Group, "Wi-Fi Peer-to-Peer (P2P) Technical Specification, Version 1.2," Dec. 2011.

[25] J. R. Eastlack, "Extending volunteer computing to mobile devices," Oct. 2011.

[26] "International Data Corporation (IDC)." [Online]. Available: http://www.idc.com

[27] "Kleiner Perkins Caufield Byers (KPCB) - Internet Trends 2014." [Online]. Available: http://www.kpcb.com/internet-trends

[28] "Geekbench 3: Cross-platform processor benchmark," Last time accessed: October 2014. [Online]. Available: http://www.primatelabs.com/geekbench/

[29] S. Schildt, F. Busching, E. Jorns, and L. Wolf, "Candis: Heterogenous mobile cloud framework and energy cost-aware scheduling," in *Proc. of IEEE GreenCom*, Beijing, China, Aug. 2013.

[30] J. Yu, E. Williams, and M. Ju, "Analysis of material and energy consumption of mobile phones in China," *Energy Policy*, vol. 38, no. 8, pp. 4135–4141, Aug. 2010.

[31] M. Y. Arslan, I. Singh, S. Singh, H. V. Madhyastha, K. Sundaresan, and S. V. Krishnamurthy, "Computing while charging: Building a distributed computing infrastructure using smartphones," in *Proc. of ACM CoNEXT*, Nice, France, Dec. 2012.

[32] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: architecture, applications, and approaches," *Wireless Communications and Mobile Computing*, vol. 13, no. 18, pp. 1587–1611, Dec. 2013.

[33] D. C. Chu and M. Humphrey, "Mobile OGSI.NET: Grid computing on mobile devices," in *Proc. IEEE/ACM GRID*, Pittsburgh, PA, Nov. 2004.

[34] K. B. Parmar, N. N. Jani, P. S. Shrivastav, and M. H. Patel, "jUniGrid: A simplistic framework for integration of mobile devices in heterogeneous grid computing," *International Journal of Multidisciplinary Sciences and Engineering*, vol. 4, no. 1, pp. 10–15, Jan. 2013.

[35] H. Xu, M. Bilec, L. Schaefer, A. Landis, and A. Jones, "Ocelot: A wireless sensor network and computing engine with commodity palmtop computers," in *Proc. of IGCC*, Arlington, VA, USA, Jun. 2013.

[36] P. Datta, S. Dey, H. Paul, and A. Mukherjee, "ANGELS: A framework for mobile grids," in *Proc. of AIMoC*, Kolkata, India, Feb. 2014, pp. 15–20.

[37] C. Shi, V. Lakafosis, M. H. Ammar, and E. W. Zegura, "Serendipity: Enabling remote computing among intermittently connected mobile devices," in *Proc. of MobiHoc*, Hilton Head, SC, USA, Jun. 2012.

[38] N. Fernando, S. Loke, and W. Rahayu, "Honeybee: A programming framework for mobile crowd computing," in *Mobile and Ubiquitous Systems: Computing, Networking, and Services*, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Springer Berlin Heidelberg, 2013, vol. 120, pp. 224–236.

[39] P. Jassal, K. Yadav, A. Kumar, V. Naik, V. Narwal, and A. Singh, "Unity: Collaborative downloading content using co-located socially connected peers," in *Proc. of IEEE PERCOM*, San Diego, CA, USA, Mar. 2013.

[40] R. Agarwal, "DRAP: A decentralized public resourced cloudlet for ad-hoc networks," Mar. 2014.

[41] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, Oct. 2009.

[42] "BOINC on Android," Last time accessed: October 2014. [Online]. Available: http://boinc.berkeley.edu/trac/wiki/AndroidBoinc

[43] "Boincoid." [Online]. Available: http://boincoid.sourceforge.net

[44] "AndroBOINC - BOINC manager for Android phones." [Online]. Available: https://code.google.com/p/androboinc/

[45] "NativeBOINC." [Online]. Available: http://http://nativeboinc.org

[46] M. Black and W. Edgar, "Exploring mobile devices as grid resources: Using an x86 virtual machine to run BOINC on an iPhone," in *Proc. of IEEE/ACM Grid Computing*, Banff, AB, Canada, Oct. 2009.

[47] "HTC Power to Give." [Online]. Available: http://www.htc.com/us/go/power-to-give/

[48] "The 10th BOINC workshop - BOINC/Android status and plans," Sep. 2014. [Online]. Available: http://boinc.berkeley.edu/trac/attachment/wiki/WorkShop14/boinc_on_android_2014.pdf

[49] A. D. Zayas and P. M. Gómez, "A testbed for energy profile characterization of ip services in smartphones over live networks," *Mob. Netw. Appl.*, vol. 15, no. 3, pp. 330–343, Jun. 2010.

[50] A. Nandugudi, A. Maiti, T. Ki, F. Bulut, M. Demirbas, T. Kosar, C. Qiao, S. Y. Ko, and G. Challen, "Phonelab: A large programmable smartphone testbed," in *Proc. of ACM SENSEMINE*, Roma, Italy, Nov. 2013.

[51] Y. Zhuang, A. Rafetseder, and J. Cappos, "Experience with Seattle: A community platform for research and education," in *Proc. of GREE*, Salt Lake City, UT, USA, Mar. 2013.

[52] C. Funai, C. Tapparello, H. Ba, B. Karaouglu, and W. Heinzelman, "Extending volunteer computing through mobile ad hoc networking," in *Proc. of IEEE GLOBECOM*, Austin, TX, USA, Dec. 2014.

[53] R. Rahman, M. Meulpolder, D. Hales, J. Pouwelse, D. Epema, and H. Sips, "Improving efficiency and fairness in P2P systems with effort-based incentives," in *Proc. of IEEE ICC*, Cape Town, South Africa, May 2010.

[54] D. Vega, R. Meseguer, F. Freitag, and S. Ochoa, "Effort-based incentives for resource sharing in collaborative volunteer applications," in *Proc. of IEEE CSCWD*, Whistler, BC, Canada, Jun. 2013.

[55] P. Darch and A. Carusi, "Retaining volunteers in volunteer computing projects," *Phil. Trans. R. Soc. A*, vol. 368, no. 1926, pp. 4177–4192, Sep. 2010.

[56] G. Huerta-Canepa and D. Lee, "A virtual cloud computing provider for mobile devices," in *Proc. of ACM MCS*, ser. MCS '10, San Francisco, California, Jun. 2010.

[57] C. Courcoubetis and R. Weber, "Incentives for large peer-to-peer systems," *IEEE J. Select. Areas Commun.*, vol. 24, no. 5, pp. 1034–1050, May 206.

[58] A. Mondal, S. Madria, and M. Kitsuregawa, "An economic incentive model for encouraging peer collaboration in mobile-p2p networks with support for constraint queries," *Peer-to-Peer Networking and Applications*, vol. 2, no. 3, pp. 230–251, Mar. 2009.

[59] G. Tan and S. Jarvis, "A payment-based incentive and service differentiation scheme for peer-to-peer streaming broadcast," *IEEE Trans. Parallel Distrib. Syst.*, vol. 19, no. 7, pp. 940–953, Jul. 2008.

[60] J. Rius, S. Estrada, F. Cores, and F. Solsona, "Incentive mechanism for scheduling jobs in a peer-to-peer computing system," *Simulation Modelling Practice and Theory*, vol. 25, no. 0, pp. 36–55, Jun. 2012.

[61] "Amazon Mechanical Turk." [Online]. Available: https://www.mturk.com/

[62] M. N. Durrani and J. A. Shamsi, "Volunteer computing: requirements, challenges, and solutions," *Journal of Network and Computer Applications*, vol. 39, no. C, pp. 369–380, Mar. 2014.

[63] T. Ghafarian, H. Deldari, B. Javadi, M. H. Yaghmaee, and R. Buyya, "CycloidGrid: A proximity-aware P2P-based resource discovery architecture in volunteer computing systems," *Future Generation Computer Systems*, vol. 29, no. 6, pp. 1583–1595, Aug. 2013.

[64] F. Costa, L. Veiga, and P. Ferreira, "Internet-scale support for map-reduce processing," *Journal of Internet Services and Applications*, vol. 4, no. 1, Nov. 2013.

[65] R. Bruno and P. Ferreira, "freecycles: Efficient data distribution for volunteer computing," in *Proc. of ACM CloudDP*, Amsterdam, The Netherlands, Apr. 2014.

[66] L. Sarmenta, "Sabotage-tolerance mechanisms for volunteer computing systems," in *Proc. of IEEE/ACM CCGrid*, Brisbane, Qld., Australia, May 2001.

[67] K. Watanabe and M. Fukushi, "Generalized spot-checking for sabotage-tolerance in volunteer computing systems," in *Proc. of IEEE/ACM CCGrid*, Melbourne, Australia, May 2010.

[68] F. Cappello, S. Djilali, G. Fedak, T. Herault, F. Magniette, V. NÃľri, and O. Lodygensky, "Computing on large-scale distributed systems: XtremWeb architecture, programming models, security, tests and convergence with grid," *Future Generation Computer Systems*, vol. 21, no. 3, pp. 417–437, Mar. 2005.

[69] S. Subashini and V. Kavitha, "A survey on security issues in service delivery models of cloud computing," *Journal of Network and Computer Applications*, vol. 34, no. 1, pp. 1–11, Jan. 2011.

[70] D. Zissis and D. Lekkas, "Addressing cloud computing security issues," *Future Generation Computer Systems*, vol. 28, no. 3, pp. 583–592, Mar. 2012.

APPENDIX A
KEY TERMS AND DEFINITIONS

Android: Android is a mobile operating system (OS) based on the Linux kernel and currently developed by Google.

Cloud Computing: Internet-based computing in which large groups of remote servers are networked in order to allow sharing of data-processing tasks, centralized data storage, and online access to computer services or resources. The term loosely define any system providing access via

the Internet to processing power, storage, software or other computing services, often via a web browser. Typically these services will be rented from an external company that hosts and manages them.

Cluster Computing: Cluster Computing consists of a set of two or more computers connected into a local area network that work together so that, in many respects, they can be viewed as a single system. Usually, computer clusters have each node set to perform the same task, controlled and scheduled by software.

Distributed Computing: The use of multiple computers networked throughout a wide geographical area, or the world via the Internet, in order to solve a single computational problem. In distributed computing, a problem is divided into many tasks, each of which is solved by one or more computers, which communicate with each other by message passing.

Embarrassingly problems: In parallel computing, an embarrassingly parallel workload, or embarrassingly parallel problem, is one for which little or no effort is required to separate the problem into a number of parallel tasks. This is often the case where there exists no dependency (or communication) between those parallel tasks.

GPU: Graphics processing unit (GPU) is a specialized electronic circuit designed to rapidly manipulate and alter memory to accelerate the creation of images in a frame buffer intended for output to a display. Modern GPUs are very efficient at manipulating computer graphics and image processing, and their highly parallel structure makes them more effective than general-purpose CPUs for algorithms where processing of large blocks of data is done in parallel.

iOS: iOS (previously iPhone OS) is a mobile operating system developed by Apple Inc. and distributed exclusively for Apple hardware. It is the operating system that powers many of the company's iDevices.

Middleware: Software that acts as a bridge between an operating system or database and applications, especially on a network.

Mobile Device: A mobile device (also known as a handheld computer or simply handheld) is a small, handheld computing device, typically having a display screen with touch input and/or a miniature keyboard and weighing less than 2 pounds (0.91 kg). Example of mobile devices are ultra-portable computers, smartphones and tablets.

Mobile Computing: Mobile Computing refers to the process of performing computation on a mobile device. In addition, mobile computing is used as a generic term describing the ability to use the technology to wirelessly connect to and use centrally located information and/or application

software through the application of small, portable, and wireless computing and communication devices.

Network Simulator 3 (ns-3): ns-3 is a discrete-event network simulator for Internet systems, targeted primarily for research and educational use. ns-3 is free software, licensed under the GNU GPLv2 license, and is publicly available for research, development, and use.

Parallel Computing: Parallel computing is a form of computation in which many calculations are carried out simultaneously, operating on the principle that large problems can often be divided into smaller ones, which are then solved concurrently ("in parallel"). In the simplest sense, parallel computing is the simultaneous use of multiple computing resources to solve a computational problem.

Smartphone: A smartphone (or smart phone) is a mobile phone that runs an operating system. Smartphones typically combine the features of a phone with those of another popular consumer device, such as a personal digital assistant, a digital camera, a media player or a GPS navigation unit. Later smartphones include all of those plus a touchscreen interface, broadband internet, web browsing, WiFi, 3rd-party apps, motion sensors and mobile payment mechanisms.

Tablet: A tablet is a mobile computer with touch-screen display, circuitry and battery in a single unit. Tablets come equipped with sensors, including cameras, a microphone, an accelerometer and a touchscreen, with finger or stylus gestures substituting for the use of computer mouse and keyboard.

Task Distribution Algorithm: In parallel computing, the task distribution algorithm represents a set of rules that are used to determine how to divide the job into computational tasks, and to find the best criteria for assigning these tasks to the different clients. The task distribution algorithm can collect and analyze different informations and attributes of the clients, as well as the user usage patterns and preferences, to efficiently manage all the available resources.

Volunteer Computing: Volunteer computing is a type of distributed computing in which computer owners donate their computing resources (such as processing power and storage) to one or more scientific computation projects. More recently, volunteer computing has moved to middleware systems that provide a distributed computing infrastructure independent from the scientific computation.