Extending Volunteer Computing through Mobile Ad Hoc Networking

Colin Funai, Cristiano Tapparello, He Ba, Bora Karaoglu, Wendi Heinzelman

Department of Electrical and Computer Engineering, University of Rochester,
Rochester, NY, USA (firstname.lastname@rochester.edu)

***Abstract*—Volunteer computing provides a practical and low cost solution to the increasing computational demands of many applications. Recent advancements in mobile device processing capabilities, combined with the energy efficiency of the mobile devices, make their inclusion in a distributed computing architecture particularly appealing. However, the intrinsic requirement of Internet connectivity to participate in volunteer computing limits the direct adoption of mobile devices due to service availability or related costs to connect to the Internet. In this paper, we propose and implement a novel computational architecture that extends the ability of mobile devices to participate in volunteer computing through ad hoc networking. By introducing decentralized task distribution points, mobile devices are invited to join the computation via device to device communication, removing the requirement for an Internet connection. Using a prototype implementation running on Android devices, we investigate the impact of a promising ad hoc communication technology, namely WiFi Direct, and two task distribution algorithms with different computation and communication overheads, under various scenarios. Experimental results show that our proposed approach is feasible with only minor additional energy consumption at the decentralized task distribution points.**

I. Introduction

In recent years, the computational requirements of various applications in domains ranging from healthcare to finance have increased dramatically. Parallel computing provides a viable solution to meeting this increasing computational demand. Various architectures, from hardware dependent solutions such as GPU programming to grid computing, have been proposed to provide a suitable parallel computing environment.

Cloud computing is an important class of grid computing, providing on demand access to high-performance computing data centers over the Internet. Such systems are essentially a cooperative group 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], that considers migrating virtual machines between physical machines in order to reduce the total power load of the data center. However, improving energy efficiency in large scale workstations is still considered a major challenge in cloud computing [4].

Instead of using dedicated hardware for parallel computing, volunteer computing aims to use underutilized 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 exceed 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 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 900,000 active participants 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. With the advancements in the area of low powered processors, mobile devices such as smartphones and tablets have become alternative computing platforms. For instance, it is not uncommon for a typical tablet such as the Asus Nexus 7 [10] to have a 1.5 GHz quad-core CPU and more than 1 GB RAM. Although the computing capabilities of mobile processors are not as powerful as the ones of a standard computer, they have been shown to be more energy efficient [11]. As a result, many traditional volunteer computing platforms have attempted to extend their operation over mobile devices. For example, Hyrax ports Hadoop Apache, an open-source implementation of MapReduce, to execute jobs on Android smartphones and, following the same approach, the BOINC project released an Android client [12] to include mobile devices in the distributed computations. More recently, GEMCloud (Green Energy Mobile Cloud) [11] has been proposed as a distributed system that exploits the energy efficiency of mobile devices to cooperatively solve computationally intensive and parallelizable tasks.

While these solutions provide a viable and energy efficient mobile computing architecture, they are based on the assumption that the mobile devices are able to communicate through the Internet with a remote cloud server, both for

This work was supported in part by Harris Corporation, RF Communications Division and in part by CEIS, an Empire State Development designated Center for Advanced Technology.

receiving the tasks and for sending back the computation results. In a rural setting where the coverage limits a mobile device's connectivity to the Internet, devices cannot participate in volunteer computing. Even in an urban area, connection to the Internet may be a limiting factor, since a wireless Access Point (AP) might not be available all the time. Although the majority of mobile devices are nowadays equipped with 3G/4G connectivity, the costs of the cellular data plan may be a deterring factor to the users participating in the computation. Hence, this dependency restricts the applicability of volunteer computing for mobile devices.

Given the above, in this paper we propose and implement a novel computer architecture that extends the existing mobile cloud computing systems by reaching mobile devices not directly connected to the Internet through ad hoc networking. In particular, we present a volunteer computing system in which a device with Internet capabilities, either WiFi or 3G/4G, can elect itself as a local task distribution point, inviting other users to join the computation via existing Device to Device (D2D) communication methods. As shown in [13] and references therein, protocols based on D2D methods have the potential to improve the spectral efficiency as well as the resource utilization for infrastructure-communication schemes, like cellular and WiFi. Starting from the existing implementation of GEMCloud [11], we design and develop an experimental system where local task distribution is performed using different D2D technologies. Besides the implementation, our focus is on the evaluation of the computing capabilities and energy efficiency of the system, as well as to prove the feasibility of the system for an existing D2D technology. In this regard, WiFi Direct [14] is considered. For the local task distribution points, two methods for distributing the computation tasks to the devices connected through D2D communication, *Batch* and *Proxy*, are also proposed. Using *Batch* mode, a set of N tasks are cached at the task distribution point and are then distributed to the ad-hoc network as they are requested. In *Proxy* mode, instead, the task distribution point acts as a gateway to the Internet-based source of the data.

The rest of the paper is organized as follows. In Section II, we present some motivating real life scenarios that would benefit from an ad-hoc cloud computing system. In Section III, we describe some background on the considered D2D communication protocol. Section IV describes our system architecture, while in Section V we discuss our experimental results, highlighting the energy consumption and evaluating the performance of the task distribution methods considered. Section VI concludes the paper.

II. Application Scenarios

Traditional mobile computing systems use the Internet as the communication architecture, as shown in Figure 1. This limits the effective use of mobile computing due to (i) the fact that some devices may not have access to the Internet for some time periods, and (ii) the cost of accessing the Internet might deter some participants from fully sharing the idle cycles of their devices. In this section, we present some reference scenarios in which having an ad hoc task distribution method will extend the total resource utilization, enabling more users to contribute their spare cycles.

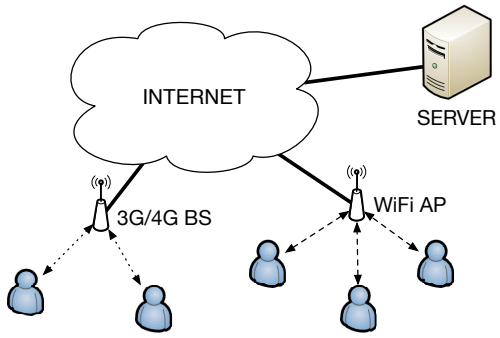


Figure 1. A traditional mobile computing topology. Users are connected to a remote cloud server through a 3G/4G Base Station (BS) or a WiFi Access Point (AP).

A. Server Driven Mobile Volunteer Computing

Most mobile devices use WiFi as the primary method of accessing the Internet due to the fact that it provides an affordable and fast connection. However, considering a user's typical mobility pattern, mobile devices do not have access to a WiFi AP during a significant portion of the day, such as time spent on transportation (e.g., buses, subways), shopping centers and remote areas not covered by WiFi hot spots. Although having a cellular data connection does allow a mobile device to participate in a mobile computing platform such as GEMCloud, the cost associated with this connection type often deters the end-users from participating.

On the other hand, during periods of non-regular mobility patterns such as vacation times, the connectivity of the devices becomes the bottleneck factor limiting the effective use of mobile computing. Although WiFi is quite widespread, Internet connectivity through WiFi hot spots generally requires a subscription fee, it is not seamless and often requires the user intervention. All these facts further limits the connectivity of mobile devices, thus decreasing their utilization in the mobile computing architecture.

A standard mobile computing system, like GEMCloud and BOINC, would not be able to utilize users who do not have an active data connection or who are not willing to use their data connection. These users would then be forced to wait until they are back "on the grid" to resume their participation.

In order to address these shortcomings, we propose the introduction of two functional roles for each client of the system: (i) task execution points (TEPs), and (ii) task distribution points (TDPs). TEPs are devices that have compatible computational platforms and are willing to participate in the volunteer computing system. TEPs compute the tasks assigned to them and send the results back to the device that requested the computation. TDPs, instead, are responsible for receiving sets of tasks from the cloud and then distributing them to the TEPs through ad hoc connections, e.g., Bluetooth or WiFi-Direct. In our proposal, a TDP could be a dedicated device that only distributes the tasks, or it could be a generous user that is willing to share its Internet connection, established either via a cellular network or through a WiFi hot spot. It is important

to note that these roles can both be taken simultaneously, and in such case, we refer to this client as a task distribution and execution point (TDEP). Therefore, rather than being fixed, the TDP and the TEP are roles that can change over time. After the computation, the task results are returned either directly to the server that requested the computation if a preferred connection becomes available, or to a TDP that will be in charge of sending them back to the requesting server. We note that the results do not have to be returned to the same distribution point that assigned the tasks.

We want to further emphasize that the proposed system could be enhanced with further economical incentives. For example, some reward could be offered to the clients acting as TDP in exchange for distributing the tasks to the TEPs that cannot receive the tasks directly. These economical incentives could form the foundation of a new business model and enhance the efficient use of computational devices. A particular scenario is the utilization of ad hoc mobile computing in public transportation. The introduction of a TDP in a bus would allow the passengers with mobile devices to participate in volunteer computing. While dedicated TDPs could be installed on public transportation, users who are willing to use their data connection can also take the role of TDP and connect the mobile devices on the bus to the cloud. In both cases, participation could be encouraged with economic incentives such as discounted bus tickets.

B. User Driven Mobile Volunteer Computing

Sometimes, the devices running distributed computing algorithms reside outside of traditional networks and away from nodes that are capable of intensive computing. For example, many applications in the area of tactical military communications, first responder networks, search and rescue operations, and sensor network operations, require computing intensive algorithms ranging from image processing to target detection that can be parallelized. However, sparsely populated remote locations frequently neither have direct access to the Internet nor are in the vicinity of other devices with access. In these situations, ad-hoc networking can be beneficial. Using a cooperative distributed computational architecture, these algorithms can be computed in a shorter amount of time using a higher accuracy and with less impact on the energy resources of critical nodes, thereby increasing the network lifetime.

One possible solution includes deploying a *Cloudlet* [15] to help accelerate battlefield applications; however, this approach creates a single point of failure where the system could collapse if the *Cloudlet* were to go down. Distributed computing addresses these problems by dividing the jobs and distributing them to the nodes in the network. The results are then collected and fused for making the final decision, thus eliminating the dependency on a specific node in the network.

Our proposed system, can be easily extended to this scenario. To this end, we introduce the additional role of Task Generation Point (TGP) to be assigned to the node that creates the jobs to be then distributed and computed by the TDPs and TEPs, respectively, present in the ad hoc network. This system can be optimized to maximize the computational power available to the nodes in need, and to maximize the lifetime of the network by letting the jobs be executed where energy resources are abundant and limiting the energy consumption on bottleneck devices.

As a final consideration, the functionality of the system presented in this section can be extended to multi-hop ad hoc networks by letting the recipients of the tasks take the role of TDPs to further distribute the tasks to the nodes around themselves. However, we limit the scope of this paper to single-hop ad-hoc networks over which the TDPs are responsible for distributing the jobs to the TEPs, and we leave the extension with multi-hop routing for future work.

III. Device to Device Communication

The computing architecture proposed in this paper is based on the availability of Device to Device (D2D) communication, which is an emerging paradigm that allows end-to-end communication between devices, without any human intervention.

Many D2D communication protocols have been proposed in the literature. These include IEEE 802.11 DCF, IEEE 802.11s, IEEE 802.11z, Zigbee, SMAC, WiFi Direct, and Bluetooth. These protocols each have their application areas.

IEEE 802.11 DCF has a basic method that provides ad hoc D2D communication and is widely available. However, it has been shown that this protocol suffers from low performance in real life implementations [16] and has a high energy consumption due to the requirement of maintaining the listening state on the radios. WiFi Direct, on the other hand, has recently been proposed to address the shortcomings of IEEE 802.11 DCF and has been designed with energy saving mechanisms leading to higher energy efficiency. IEEE 802.11s and IEEE 802.11z add mesh networking and direct communication on top of the IEEE 802.11 family. These additions are only relevant in a multi-hop network. SMAC and ZigBee are energy efficient protocols that are designed for sensor networks. The data rates supported by these protocols are very low and thus not suitable for large tasks. Moreover, their availability is limited to low power sensor nodes that have very low computational capacity.

Given the above, in this paper we focus our attention on WiFi Direct. This is because we believe that WiFi Direct is the most promising technology, and it is commonly available off the shelf in several mobile devices[1]. In the remainder of this section, we present a brief overview of this protocol.

A. WiFi Direct

WiFi Direct [14] is a standard released by the WiFi alliance that enables D2D communication without requiring a wireless AP. During D2D communication, devices form a group were one of them is the Group Owner (GO) and all the others are considered Group Members (GMs). It is important to note that these roles are not fixed and can change dynamically. Additionally, WiFi Direct groups can also consist of nodes that do not support WiFi Direct, but do support the IEEE 802.11 standard that the group is operating. WiFi Direct utilizes IEEE 802.11 a/b/g/n infrastructure mode, and can transmit either at

[1]Since Bluetooth is an incumbent technology widely deployed, we consider its integration in our system as a future work.

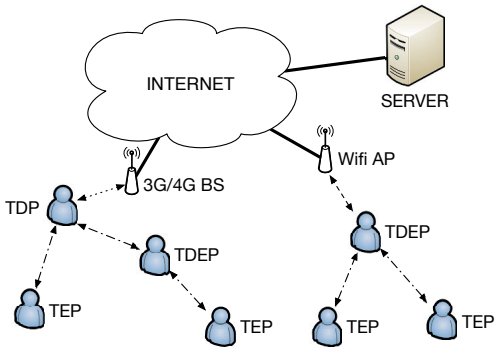


Figure 2. Example of ad hoc mobile computing topology with corresponding functional role assigned to each of the clients. Task distribution point (TDP), task execution point (TEP), and task distribution and execution point (TDEP).

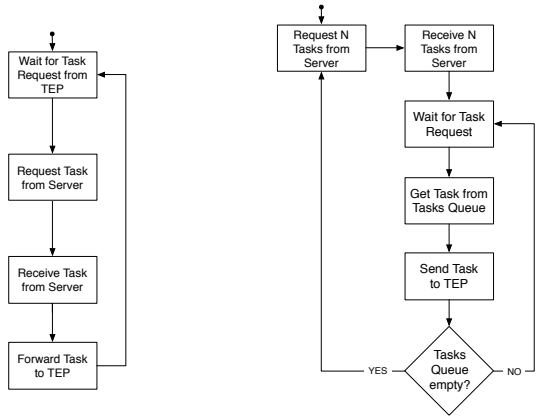


Figure 3. Task distribution flowcharts for the TDP operating according to the *Proxy* (a) and *Batch* (b) methods.

2.4GHz or 5GHz. Since WiFi Direct utilizes the IEEE 802.11's *infrastructure* mode, all of the QoS, power saving, and security protocols of IEEE 802.11 are also inherited. Similar to a typical IEEE 802.11 network, where nodes find and connect to APs, the GO also acts as a soft AP, advertising and allowing nodes to join the group. Group advertisement is performed using beacon packets, just like a typical IEEE 802.11 AP, and the GO is responsible for giving control of the channel to nodes in its network as well as routing data through its group. Nodes that support WiFi but do not support WiFi Direct can still connect to a WiFi Direct group [14] [17] and are referred to as *legacy clients*. GMs are the nodes that support WiFi Direct and hence can capitalize on the WiFi Direct power saving options. The nodes that support WiFi Direct go through a group formation process in order to determine the roles of the GO and the GMs.

IV. System Architecture

The traditional mobile computing platform, as considered in [11], [18] and [19], is presented in Figure 1.

Our goal is to extend the range of computation devices of a traditional mobile computing system via ad hoc communications. In Figure 2, we show an example of this type of system, where certain users are allowed to become local Task Distribution Points (TDPs and TDEPs in Figure 2) and provide access to the distributed computing system to other clients, using an ad hoc communication method. Thus, each TDP is in charge of requesting tasks from the server, and then it distributes these tasks to its clients. Moreover, the TDPs are responsible for organizing the results, resolving any failures, and returning the results back to the remote server.

The general idea behind our approach is similar to the one proposed in [20], that considers a manager/worker model, in which the manager receives requests from the workers and responds to these requests by assigning tasks.

A. Task Distribution Point

In our system, a central role is played by the devices that act as TDPs and, for this reason, the way in which the TDP operates will affect the performance of the complete system. In particular, the TDP is required to receive tasks from a remote server, distribute the tasks to the TEPs through D2D communication, receive the results of the computations from the TEPs and then send the results back to the remote server.

The TDPs can receive the tasks from the server following different approaches. In general, the different methods can be classified based on the time at which the actual request happens: 1) following a proactive approach, meaning that the task requests to the server are made before the actual requests from the TEPs, or 2) following a reactive approach, thus requesting the task from the server at the time of the actual TEPs requests. Following a similar idea, the TDP can send the results back to remote server either immediately, at the time of the reception from the TEP, or delayed, after collecting multiple results. To this end, we propose two different TDP modes of operation: a simple *Proxy* method, that considers a reactive task distribution technique and an immediate forwarding of the results, and a more involved *Batch* approach that, instead, implements a combination of proactive task requests with a delayed results transmission.

Proxy: This method represents a basic mode of operation and requires little intelligence to be added to the device that will act as TDP. In particular, a *Proxy* TDP acts as a gateway and forwards all requests and responses directly between the remote server and its clients. Figure 3(a) shows the flow diagram of the *Proxy* task distribution procedures, while Figure 4(a) presents the flow diagram relative to the collection of the results.

Batch: In this method, the TDP proactively requests a set of N tasks from the server and caches them so that, when it receives a request from the TEPs, the TDP will promptly respond with one of the cached tasks. After completing the task, the TEP returns the result to the TDP, and the TDP stores these results before sending them back to the server. Figure 3(b) shows the flow diagram of the *Batch* task distribution procedures, while Figure 4(b) presents the flow diagram relative to the collection of the results.

B. Task Execution Point

In the proposed system, we assign the role of task execution point (TEP) to all the devices that participate in the distributed

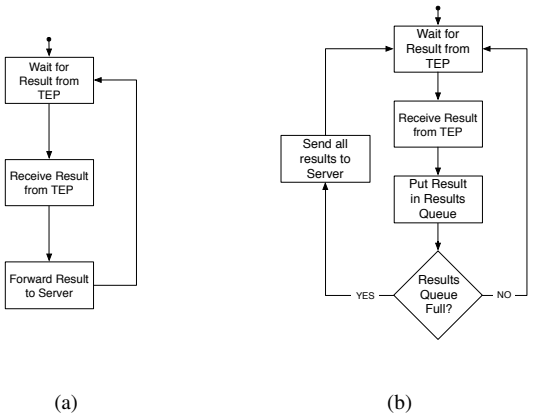


Figure 4. Result collection flowcharts for the TDP operating according to the *Proxy* (a) and *Batch* (b) methods.

computation, disregarding the connection used for receiving the task and sending the results of the computation. Thus, a traditional client of the volunteer mobile computing system that receives the tasks directly from the server and a client connected through D2D communications are both considered TEPs. Moreover, the way in which the TEPs receive tasks and send results back to the server are completely transparent, and the protocol used for the communication is exactly the same in both cases.

C. Complexity Considerations

The objective of our work is to extend the volunteer mobile computing architecture by introducing local TDPs, whose operation is completely transparent to both a standard client (i.e., TEP) and the remote server. Thus, the additional complexity of our system when compared to an existing mobile computing architecture, resides on the requirements of the D2D communication technology, on the ability to advertise and discover local TDPs, and on the intelligence necessary for the local task distribution procedure.

In particular, the *Proxy* implementation simply forwards all the requests to and from the server to the ad hoc network clients. Thus, a *Proxy* TDP acts as a local gateway, and it performs a translation between the Internet interface and the D2D domain. We consider the *Proxy* method to be the simplest mode of operation that can be implemented in a TDP. It simply forwards every communication to and from the GEMCloud server, thus introducing additional delay due to the additional communication hop. However, it is considered as a reference implementation for evaluating the performance of other task distribution techniques. The *Batch* task distribution approach, instead, overcomes the inefficiency in the network utilization of the *Proxy* method by caching a set of N tasks at the TDPs for faster distribution to the local TEPs. Moreover, a *Batch* TDP stores a set of M results before sending them to the server. Thus, some additional computation and data storage is required in order to handle the task requests and the caching operations.

In section V we provide some experimental results that show the impact in terms of time delays and energy consumption of the proposed task distribution methods.

D. Implementation

Given that the idea behind our work is to enhance the performance of a mobile computing system, we extend the GEMCloud [11] architecture to support TDPs that can distribute tasks to TEPs connected through D2D communication. GEMCloud provides us with a platform, and an Android application, to evaluate the benefits of extending volunteer computing through ad hoc networking.

An overview of GEMCloud's architecture, is presented in [11]. In GEMCloud, the server acts as a central point by coordinating and distributing tasks to the connected clients. The client connects to the server using either 3G/4G or WiFi. After connecting, the server checks to see if the client requires an updated version of GEMCloud [11]. The server then authenticates the client using a set of parameters that are exchanged and, after that, it assigns tasks to the client. The tasks are stored in a list, and they are assigned in a first in first out order. Each of these tasks are independent and do not need to be returned, but the quality of the final results will depend on the number of tasks that are actually returned after the distributed computation.

In our implementation of the *Proxy* and *Batch* distribution methods, the server is the same as in GEMCloud, and the client takes the role of TEP. Using our system, we can extend the devices that can participate in the GEMCloud computations by connecting additional TEPs through the TDPs. The *Proxy* task distribution method provides operation very close to GEMCloud: the TDP behaves as a proxy, and it forwards any requests between the TEP and the server. In the *Batch* implementation, instead, the TDP requests and caches a set of tasks from the server. The TDP still goes through the same process as in GEMCloud, where the server checks for updates and authenticates before sending a task; however, the TEPs that connect to a TDP running our *Batch* implementation will have their version checked against the local version that the TDP is running. The TDP then will send tasks from its tasks queue. It is important to note that in both task distribution methods, in addition to the role of a TDP, the device may also act as a TEP. In the *Proxy* method, devices that serve as TDPs connect and request tasks from the server also for themselves, similar to the original GemCloud operation. In the *Batch* method, devices acting as TDPs take tasks from their own queue rather than requesting them from the server. In other words, these devices can be modeled by combining the TDP and TEP roles.

V. Performance Evaluation

As presented in Section IV, our system is composed of four elements, namely TEP, TDP, server and database. Since we assume that the TDP-TEP interactions are transparent to the remote server and database, and that their operations are the same as traditional cloud computing systems, in our performance evaluations we focus on the performance of the TDPs and TEPs.

A. Experimental Setup

In order to evaluate the performance of our proposed system we used five Asus Nexus 7 (N7), 2013 edition [10], which are equipped with a Qualcomm® Snapdragon™ S4 Pro 8064 Quad-Core, 1.5 GHz CPU, 2 GB of memory and 16 GB of storage. All the N7 are running Android 4.4.

The tasks considered in this paper implement a distributed algorithm for the prediction of the structure of proteins [21]. This algorithm consists of a combination of C++ code and a set of supporting scripts that are handled through the GEMCloud application. In particular, the task assigned to the TEP is a 2 B random number that represents a sequence of amino acids, while the result of the computation is a set of predicted structures for the given sequence. The result data size is 9 KB.

We measure the devices' power consumption using the "Watts up? PRO" power meter [22]. According to its specifications, the meter has an accuracy of $\pm 1.5\%$ in terms of wattage measurement. In order to achieve a good sampling rate for our measurements, we set the recording interval of the meter to its lowest value of 1s. We measure the device's energy consumption while the screen is off and using the Asus wall charger provided with the devices. Moreover, all the results presented in this section are obtained by averaging, for each experiment, the results of 10 runs. We allowed for one N7 to be connected to the Internet through a legacy WiFi AP and then to act as a TDP for the other N7, by using both the *Batch* and *Proxy* task distribution methods described in Section IV-A. The TDP ad hoc communications are handled with WiFi Direct, as highlighted in Section III.

Moreover, we allowed the TDP to additionally act as a TEP, thus taking part in the computation of the received tasks. Finally, for the *Batch* distribution method, we set $N = M = 10$.

One of the features of GEMCloud is to let the user decide on the maximum amount of CPU resources assigned to computation. Since our prototype application is derived from GEMCloud, we inherited this functionality and set the maximum CPU resources assigned to computation to 25%. This means that for the TDP, both the calculation and the distribution could only take up 25% of the CPU's resources.

B. Numerical Results

We start our performance evaluation by comparing the impact of the different functional roles on a single device. To this end, in Figure 5, we provide the time and relative energy consumptions, required for a single device to compute a set of 50 tasks when it is operating as a standard GEMCloud client, as a *Proxy* and *Batch* TDEP or as a TEP connected using WiFi Direct to a *Proxy* and *Batch* TDP. We note that for the TDEP, we do not consider the presence of additional TEPs, in order to only evaluate the impact of advertising the task distribution service. As expected, the results in Figure 5(a) show that the total time required for computing the tasks is similar for all the devices directly connected to the remote server and for the TEP connected to a *Batch* TDP. Conversely, the time is slightly higher for the TEP connected to a *Proxy* TDP because it requires an extra communication hop in order to receive

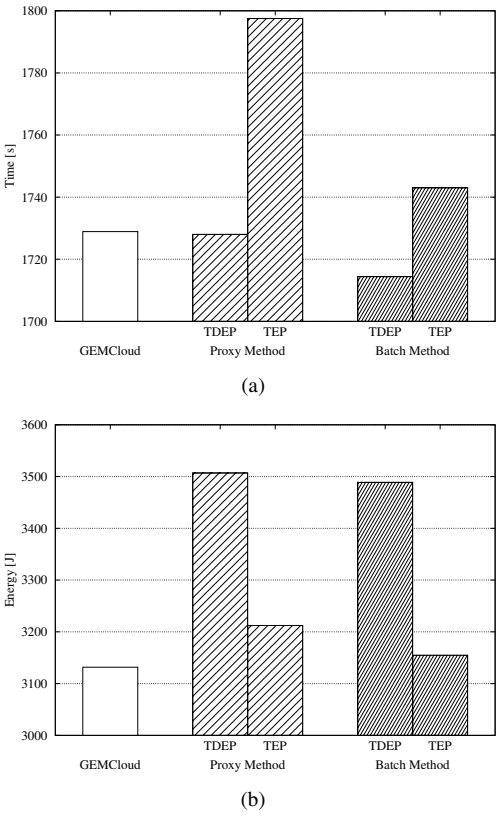
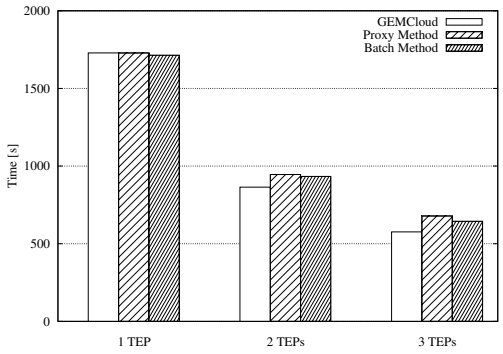


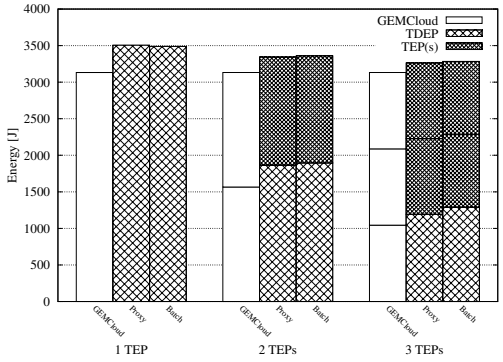
Figure 5. Time (a) and energy consumption (b) for computing 50 tasks, for a systems where only 1 node is allowed to execute the tasks. "TEP" refers to the the performance of a TEP that receives the tasks from a local TDP.

the tasks and send the results to the server. Moreover, by comparing the energy consumptions of the TDEP and TEP with that of the GEMCloud client (see Figure 5(b)), we note that the task distribution advertisement has only a small impact on the devices (around 12%). At the same time, the TEP connected to both a *Proxy* and *Batch* TDP requires an amount of energy comparable to the one required by GEMCloud.

In Figure 6, we compare the performance of a system where multiple devices participate in the computation of 50 tasks. In particular, we present the time (Figure 6(a)) and the energy consumption (Figure 6(b)) required by GEMCloud and by our proposed system, operating according to the *Proxy* and *Batch* task distribution method, when the execution of the tasks is divided between 1, 2 and 3 devices. We note that the best performance, in terms of both time and energy consumption, are achieved by GEMCloud. Nevertheless, our proposed system introduces only a small time delay and energy overhead due to the local task distribution method. The energy consumption of the TDEP is slightly greater than those of the TEPs. This is due to the overhead introduced by the task distribution process but also related to the fact that, when assigning the task, the TDEP prioritize itself over the TEP(s) connected through it (the motivation behind this design choice is that the TDEP provides an higher reliability in sending the result of the task execution, since it is directly connected to the remote server). Thus, the average number of tasks computed



(a)



(b)

Figure 6. Time (a) and energy consumption (b) for computing 50 tasks, when 1, 2 and 3 nodes are allowed to execute the tasks. For the proxy and batch methods, the case "1 TEP" refers to a system with one TDEP. For multiple TEPs, the contributions of the different devices is also presented.

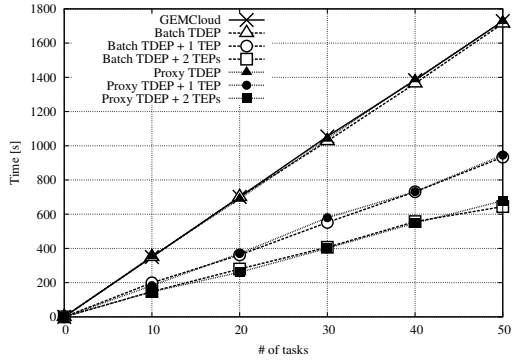


Figure 7. Computational time vs. number of tasks to be computed, for a system where only 1 node is able to connect to the Internet.

by the TDEP is higher (27.1 for 2 TEPs and 17.8 for 3 TEPs), which in turns translates in a greater energy consumption. Moreover, we note that the energy overhead at the TDEP is constant for increasing number of TEPs.

Finally, in Figure 7 we evaluate the execution time as a function of the number of tasks to be computed, when we can exploit the presence of local TEPs without Internet connectivity. In particular, we consider that only one device is able to connect to the remote server, either as a standard GEMCloud client, or as a TDEP that then is able to distribute the tasks via WiFi Direct. As expected, allowing for additional TEPs to participate in the computation reduces significantly the overall time required for the computation of the tasks.

VI. Conclusions

In this paper, we proposed and implemented in Android a novel computational architecture that extends the range of devices participating in volunteer computing through ad hoc networking. Experimental results prove the feasibility of the technique when considering WiFi Direct as a D2D communication technology, and show that the additional energy consumption required at the task distribution point is small. The promising results of this paper motivate future extensions with multi-hop routing.

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, June 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. 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 Comput.*, vol. 5, no. 3, pp. 88–95, May 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] D. P. Anderson, "Volunteer computing: The ultimate cloud," *Crossroads*, vol. 16, no. 3, pp. 7–10, Mar. 2010.

[10] "Asus Nexus 7 (2013)," Last time accessed: August 2014. [Online]. Available: http://www.asus.com/Tablets_Mobile/Nexus_7_2013/

[11] 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.

[12] "BOINC on Android," BOINC's Homepage, Last time accessed: August 2014. [Online]. Available: http://boinc.berkeley.edu/

[13] A. Asadi, Q. Wang, and V. Mancuso, "A survey on device-to-device communication in cellular networks," *ArXiv e-prints*, Oct. 2013.

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

[15] T. Soyata, R. Muraleedharan, S. Ames, J. Langdon, C. Funai, M. Kwon, and W. Heinzelman, "Combat: mobile-cloud-based compute/communications infrastructure for battlefield applications," in *Proc. of SPIE*, Baltimore, USA, Apr. 2012.

[16] G. Hiertz, S. Max, Y. Zang, T. Junge, and D. Denteneer, "Ieee 802.11s mac fundamentals," in *Proc. of IEEE MASS*, Pisa, Italy, Oct. 2007.

[17] D. Camps-Mur, A. Garcia-Saavedra, and P. Serrano, "Device-to-device communications with Wi-Fi Direct: overview and experimentation," *IEEE Wireless Commun.*, vol. 20, no. 3, pp. 96–104, June 2013.

[18] E. Chen, S. Ogata, and K. Horikawa, "Offloading Android applications to the cloud without customizing Android," in *Proc. of IEEE PerCom*, Lugano, Switzerland, Mar. 2012.

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

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

[21] J. J. Ellis, F. P. Huard, C. M. Deane, S. Srivastava, and G. R. Wood, "Directionality in protein fold prediction," *BMC bioinformatics*, vol. 11, no. 1, p. 172, Apr. 2010.

[22] "Watts up? PRO Watt meter." Last time accessed: August 2014. [Online]. Available: https://www.wattsupmeters.com/secure/products.php?pn=0