Towards Distributed Mobile Computing

Giuseppe Massari
Dipartimento di Elettronica,
Informazione e Bioingegneria
Politecnico di Milano
Milan, Italy
Email: giuseppe.massari@polimi.it

Michele Zanella
Dipartimento di Elettronica,
Informazione e Bioingegneria
Politecnico di Milano
Milan, Italy
Email: michele.zanella@mail.polimi.it

William Fornaciari
Dipartimento di Elettronica,
Informazione e Bioingegneria
Politecnico di Milano
Milan, Italy
Email: william.fornaciari@polimi.it

***Abstract*—In the latest years, we observed an exponential growth of the market of the mobile devices. In this scenario, it assumes a particular relevance the rate at which mobile devices are replaced. According to the International Telecommunicaton Union in fact, smart-phone owners replace their device every 20 months, on average. The side effect of this trend is to deal with the disposal of an increasing amount of electronic devices which, in many cases, are still working. We believe that it is feasible to recover such an unexploited computational power. Through a change of paradigm in fact, it is possible to achieve a two-fold objective: 1) extend the mobile devices lifetime; 2) enable a new opportunity to speed up mobile applications. In this paper we aim at providing a survey of state-of-art solutions aim at going in the direction of a Distributed Mobile Computing paradigm. We put in evidence the challenges to be addressed in order to implement this paradigm and we propose some possible future improvements.**

Keywords—Mobile applications, Distributed computing, Mobile computing, Distributed systems, Parallel processing, Pervasive computing, Middleware, Software systems.

I. Introduction

Today mobile and personal devices are becoming more pervasive than ever. Moreover the hardware and the computing resources currently available on such devices make them capable of executing performance-hungry multi-tasking applications.

A worth to be considered indication comes from the International Telecommunication Union. According to it, the average rate at which smartphones are replaced by the respective owners is around every 20 months. The consequence of this is an increasing amount of unused devices forgotten in our drawers or destined to disposal.

In this paper, we aim at focusing on the fact that this multitude of mobile devices can be still exploited, for instance to run “background” tasks. Despite their different hardware capabilities (e.g. screen resolution, camera quality, sensor availability, ...) in fact, these devices can share the same operating system (e.g. Android), such that it is possible to have the same software stack on a wide variety of devices, allowing us to seamlessly deploy applications on a device or another. These considerations lead us to think that a set of mobile devices can be potentially exploited as a *distributed computing system*, where each node is represented by a device and the interconnection infrastructure relies on wireless communication technologies. In a scenario like this, we could start addressing problems like defining a novel programming paradigm, and we could design an inter-device run-time layer in charge of handling the placement of the application tasks in a distributed fashion [19], [33], [62].

However, in such a scenario we should also take into account the typical constraints of mobile devices, concerning thermal and energy management. In this regard, a distributed rearrangement of mobile devices can lead to improvements in terms of energy efficiency and heat dissipation, by means of *task offloading* and dynamic *load balancing*.

Of course, there are some practical challenges and technical issues that we need to overcome in order to effectively exploit devices in a distributed manner. First, as already said, we must introduce a specific programming paradigm. Second, the device management requires a distributed strategy that takes into account the heterogeneous distribution of capabilities and current battery status over all the available devices. Last but not least, we must guarantee consistent executions, given the unreliable nature of battery-powered devices [36], [80], without breaking security.

The paper is structured in two main parts: the first, covered by Section II, discusses the main approaches related to the mobile distributed computing, while the second one, covered by section III, introduces the main techniques and implementations focusing on task offloading in mobile contexts. Section IV exposes briefly how we can introduce resource management strategies in the overall picture, in order to achieve the aforementioned benefits. Section V discusses about the direction of future researches and conclusions.

II. Mobile Distributed Computing Approaches

In recent years early studies began to explore the idea of implementing the distributed computing paradigm in systems based on mobile devices [21]. The approaches and depicted scenarios are however quite different from each other.

The wireless network improvements lead researchers to create distributed systems that cooperate in computational-intensive tasks and to coin some paradigms. For example, in the *opportunistic computing* paradigm [17], [16], mobile devices are connected in an ad-hoc local wireless network to take advantage of the computing resources of other devices. The most recent proposal in this direction is the AnyRun Computing (ARC) [27] system, which dynamically selects the best device for offloading the execution of tasks.

In what follows, we present some recent and noticeable solutions for mobile systems that exploit some well-known distributed infrastructures and approaches. They can be divided mainly in five categories, based on the computing paradigm they exploit:

- Transparent Computing
- Flexible Computing
- Voluntary Computing
- Enterprise Computing
- High-Performance Computing

It is also common to find service-oriented architectures as they enable an high level of transparency for the user, and they are easily extensible with new services and interfaces. An example of a service-oriented approach can be found in [75]. The authors proposed a Web Service Initiation Protocol (WIP) integrated in Android, making the device a web service SOA-based platform with real-time communication capabilities. This solution uses a proprietary application, the 2SAP, to perform the service discovery and registration. It does not allow to include a developer-extension of the application, and the services can be managed only through the given application proxy. Moreover, the solution does not consider the availability of computing resources and energy budget, because the exposure of the service is not linked to the capabilities of the device.

A. *Transparent Computing*

A service-oriented approach that must be aware of the system resources is presented in [77], where the concept of transparent computing [79] is transposed to the mobile world. Its goal is to provide users with transparent services: users only concern whether they can get the service or not, but without any need to know the underlying details. To do this, the solution requires a lightweight terminal without an operative system installed in advance. The software stack, including the operating system, the applications and the data is downloaded from a remote server as a virtual machine, on the basis of the user requirements and the computing resources availability. In terms of security there is also the possibility of introducing different authority provisioning to different resources and services. Unfortunately, this approach requires the devices to have an Internet connection. Furthermore, there is no mention of an energy-aware run-time management of the device.

The distributed aspect of this approach is given by the possibility of having the same services available for different terminals. However this is not properly what we intend as "distributed computing", since the computation is not distributed among devices but it is locally performed on the current device, once the service or the application is loaded.

B. *Elastic Personal Computing*

Based on the concept of *Flexible Computing* [60] this paradigm takes advantage of the interconnected devices, relying on the fact that in many cases processing data in-place and exchanging them directly between devices can overcome bandwidth limitations, hence resulting in a more efficient approach with respect to offloading the entire job to a remote server.

Daz-Sanchez et al. [22] proposed the Light Weight Map Reduce (LWMR) framework to enable the possibility of submitting a job by any device or group of devices, collecting the outcome and delegating tasks to other devices upon battery, network or location changes. This refines the Elastic Computing concept exposed in [70], providing a mobile version of the Hadoop MapReduce framework.

Similarly, the Hyrax system [54] implemented the job distribution mechanism by porting Hadoop to interconnected Android devices. The centralized-architecture limitations of Hyrax are then overcome by MC^2 [40], which makes possible the setup of personal cloud computing systems made by nearby mobile devices. MC^2 provides the possibility to create a private or public cloud service.

Elespuru [25] instead, investigates the feasibility of using smart mobile devices in a MapReduce system. The author implements a client-server MapReduce system for mobile devices, which shows that the devices are capable of performing at roughly an order of magnitude more slowly than the traditional clients, demonstrating that a large portion of processing can be moved to them, if many enough exist at a given time to perform the necessary workload.

Finally, it is worth to mention GEMCloud [4], whose purpose is to exploit mobile devices to execute computationally intensive and parallel tasks with a high degree of energy efficiency. The system is made by a central server and a database, in charge of discovering available devices onto which deploying tasks. On the device side, a client application makes the devices visible or not, according to the device status, e.g. CPU and memory usage, battery level, and running applications. However, what is still missing in this solution is the possibility of implementing a scheduling and task placement policy, aiming at maximizing performance or minimizing energy consumption.

C. *Volunteer Computing*

The so-called *volunteer computing* [1] paradigm has been introduced in 1996 by Luis F. G. Sarmenta. In this approach users make their devices available for hosting external computational by intensive tasks. It became more and more attractive for the users so that some projects received considerable media attention, such as SETI@home [2] and Folding@home [29].

The most representative framework enabling this paradigm is BOINC [10], started as a project for researchers to exploit the processing power of personal computers around the world. It has been extended by the NativeBOINC project [66] for Android devices. Similarly, [24] links the BOINC middleware and the concept of volunteer computing to the mobile world.

In [32] the volunteer computing paradigm has been extended to the mobile devices not connected to Internet, exploiting *WiFi Direct* to setup point-to-point connections. The device (node) can therefore become a distribution point or a simple proxy node towards Internet. The main goal of the solution is to extend the task distribution network, with an eye on the device applications and resources management.

Another extension of the volunteer computing paradigm is REPC [23], a generic "randomized" task assignment framework that exploits mobile devices for participatory computing.

REPC is another example of server-based distributed system. A centralized server in fact, hosts the execution of a Task Manager, in charge of assigning tasks to subscribed devices. The overall goal is to guarantee the completion of a given minimal number of tasks, minimizing the number of tasks assigned per device. The central server is involved also in the estimation of the run-time statistics regarding the tasks execution.

D. Enterprise Computation

The idea of using mobile devices for distributing the computational load has found interest also in the enterprise world. Arslan et *al.* in [3], [55] proposed a distributed computing infrastructure using smartphones in enterprise context. The main idea is to use employees enterprise device to perform the computation while they are recharging, instead of the company's servers, in order to reduce energy consumption and costs. Although the solution is quite complete and takes into account also the device computation capability and power status, the application context is closely linked to enterprise workloads. Moreover, the client-server architecture represents a limitation in terms of scalability and flexibility. Another lack of this solution is the fact that the computation capability of the device is evaluated by the server by estimating the task completion time from previous task executions. It does not rely instead on a resource manager instance, running on the device that can expose capabilities and perform local optimizations, taking into account also the workload launched on the device by the user. Moreover, the recharging status is the only condition for which the mobile device is considered available.

E. High-Performance Computing (HPC)

The High Performance Computing (HPC) area had also considered mobile devices as computational nodes of a parallel system. In this regard, the DroidCluster proposal [12] proved to be feasible with collaborative Android systems by using the Message Passing Interface (MPI) as reference programming paradigm. DroidCluster claims to be non-invasive, i.e., the framework does not interfere with the devices primary function. Obviously, in this case the target workload is made by parallel HPC applications, and no resource management is considered.

F. Experimental Results and Limitations

Analyzing the previous works we noticed that they do not consider some aspects of the devices that could be interesting to study in depth. First of all they do not perform resource and energy management, and optimization at single device level: [3], [23] and [4] are the only that consider device resource capabilities, even if in the first work the estimation is done remotely by the server basing on the previous completion time of the tasks, whereas in the second one it is delegated to a local ”passive” application. In future works it will be interesting to investigate how a local resource manager affects resources utilization and remote server decision.

A local resource manager could also act as a decision-maker for device selection during tasks distribution, monitoring the device resources and energy, and performing an optimal

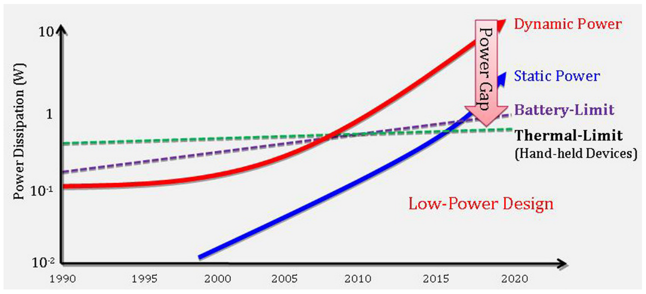

Fig. 1. The graph shows the growth of power dissipation and the battery limit (ordinate-axis) over years.

task distribution to other devices or servers. Moreover most solutions rely on a single device that acts as a ”supernode” and makes decisions, while recently research is moving on to investigate a multi-agent context where nodes cooperate and make decisions in a fully distributed way.

Finally, no previous solutions consider the management and optimization of Android applications, but mainly HPC applications already designed and implemented for multi-node parallel execution. Future research is going to concentrate on studying in depth the possible interactions between HPC and device applications, and their management strategies.

III. Computation Offloading for Mobile Systems

In this section we expose many issues and solutions related to computation offloading. The *offloading* is a practice studied from the mid 1990s, consisting of executing part of an application (a specific task) on a secondary computing device or system.

Since, as shown in Figure 1, the energy budget management is still a challenging activity, a possible exploitation of task offloading can be the pursuing of a device lifetime extension goal, by scheduling the task execution on a remote server providing higher performance.

Before going through the various solutions, it is necessary to clarify the difference between offloading practice and other type of distributed computing approaches. As exposed by [46], offloading is not a classic client-server architecture because the device is not a thin client and it can decide whether delegating the computation to the server or doing it locally. Moreover, it differs from the load balancing techniques used in grid computing because with the former the offloaded tasks are migrated to servers that are not in the same computing environment. An exception can be found by exploiting mobile distributed computing, because in this case the offloaded tasks are migrated to other personal devices.

Looking at the research since nineties onward, we can group literature works in three different categories: feasibility studies, decision algorithms and framework proposals. Regarding the first category, most of the feasibility studies have been proposed before the year 2000 [8], [30], [41], [43], [57], [61], [71], [73], [74]. Since 2000s onward, more works on policies and decision algorithms started to appear. The proposed algorithms aim at managing or adapting the execution of the mobile applications, on the basis of three different approaches: program partitioning [14], [28], [47], [48], [56], [67], [69], [76], [78], energy-aware execution [63], [65] and context-awareness [35], [37], [38], [72]. A policy for task offloading decision is typically driven by objectives of *performance improvement*, i.e., meeting response time requirements, meeting real-time constraints, etc…or *energy saving*, i.e., extending battery life, achieving green-computing purposes. Recently, some frameworks for mobile computing offloading have been proposed and have gained attention. In what follows we discuss about some implemented tools that enable the computation offloading. Mainly offloading is performed on cloud-based systems, exploiting the computational power of high performance servers, although some recent works aim at exploiting an interconnected mobile device as a cloud system: probably this will be the future research scenario. Among the various technological improvements that contributed to the growth of offloading interest, there are wireless network bandwidth increasing, mobile agents development and the rebirth of virtualization [46].

Regarding the development of Mobile Cloud Computing-applications, Orsini et al. [59] identified six main requirements: *availability*, *portability*, *portability*, *scalability*, *usability*, *maintainability* and *security*. Among these, to reach the scalability requirement, one of the most used approach is to partition the application into tasks of different granularity level, although it involves some major difficulties such as:

- *Correctness*: identifying which parts of the application have to run on the local device;
- *Effectiveness*: avoiding that a network delay caused by offloading is greater than the reduced execution time;
- *Adaptability*: offloading adaptation to applications user requirements and run-time environment changes.

Application partitioning and offloading can be performed at class or method level to obtain a more fine-grained resource allocation. Table I shows further analyzed tools and their corresponding partitioning level. It can be noticed that works originally concentrated at VMs and class level, while in recent years they exploited the method-level partitioning both relying on programmer annotations or on automatic partitioning tools [50]. In what follows, we will provide a taxonomy partition of some noticeable works that perform application partitioning and offloading.

A. Server-based Architecture

The first noticeable work that handles application partitioning we can find is J-Orchestra [67]. It is a tool that partitions any Java application at bytecode level, providing a GUI to ask programmers to manually select the “offloadable” classes, but it does not consider mobile applications and devices.

In 2010 Cuervo et al. presents the MAUI system [20], a server-based system that enables a fine-grained energy-aware offload of mobile code to a server infrastructure. It embraces the programmer-guided partitioning approaches in which the programmer can annotate the method that can be offloaded for remote execution. MAUI has a client-server architecture: the server acts as a coordinator for the offloading decision and local resources management for the incoming request. Four important goals are reached by this work: code portability

TABLE I. OFFLOADING TOOLS PARTITIONING LEVEL

Year	Paper	Tool name	Partitioning Level
2002	[67]	J-Orchestra	VM
2004	[64]	Cloudlets	Class
2010	[20]	MAUI	Method
2011	[53]	μcloud	Class
2011	[15]	CloneCloud	Thread
2012	[44]	MACS	Class
2012	[34]	COMET	Thread
2012	[78]	DPartner	Class
2012	[42]	ThinkAir	Method/VM
2012	[13]	N.A.	Method/VM
2016	[27]	ARC	Method
2016	[52]	COMPSs-Mobile	Method

exploiting partitioning for .NET applications, programming reflection, type safety and method profiling through serialization to determine its network shipping costs. However, it is not considered the real scalability and usage of the cloud infrastructure.

In [78] the authors proposed an automatic tool (DPartner) to refactor the Android application bytecode to enable task offloading. The tool performs various static analysis on the application bytecode, detecting which classes can be considered “movable” among different devices, through a novel *proxy and endpoint* software design pattern (Figure 2). The refactored application package file is then deployed in form of a Java archive file for the device, and movable Java bytecode classes for the remote server. The decision about offloading classes is also based on a static analysis of the performance and power consumption of the bytecode execution [9]. The endpoint then is in charge of monitoring the top n computational intensive classes, executing a prediction algorithm and handling the actual communication between classes. A major goal reached by this work is the fact that, unlike ThinkAir or MAUI, it is transparent to the developer and it does not require any input or environmental conditions as CloneCloud does.

Eom et al. [26] presented MALMOS, a machine-learning task offloading scheduler. This framework enables a dynamic adaptation of the scheduling decisions based on the observation of the correctness of the previous offloading decisions. In this work, the authors integrated their system with the DPartner partitioning tool, improving the need of user-input and static decision rules previously required by DPartner. The major contribution is that, albeit some offloading tools retrieve dynamically the status of the device, however they depend on static pre-defined rules, application pre-processing or user-input to make decisions on whether and where to offload.

B. Cloud-based Architecture

The main difference between cloud-based and server-based solutions is that the former use remote cloud services that are not necessarily in the same environment of the device. While cloud-based systems are highly scalable, server-based (or cloud-free) architectures can count on the local proximity of the servers, avoiding network latency, but lacking of flexibility because of the need of dedicated computing resources without the possibility of dynamically scaling the system. The Cloudlet model coined and presented by Satyanarayanan et al. in 2004 is one of the first works that exploit a cloud-based architecture. Cloudlet is a mobile small-scale cloud datacenter that is usually located closer to the personal device environment than common cloud infrastructures (i.e. in the same Local Network). The study shows the limitations of a WAN-based solution from the perspective of the communication latency impacting on the usability and responsiveness of mobile applications. The solution uses a VM approach over target devices to encapsulate and separate the guest software environment from the cloudlet host.

In [15] it is introduced the CloneCloud framework. It is a cloud-based tool extracting binary pieces of a given process to be potentially executed on a virtual smartphone clone. The clone would run on a cloud to speed-up the overall process execution. Unlike MAUI, CloneCloud does not require developer help or code annotation, because it performs an offline static analysis of different running conditions of the binary on both the target smartphone and the cloud. The outcome of this analysis populates a database composed by the pre-computed partitions of the binary that should be migrated. The main lack of this approach is the need of input and environmental conditions, and consecutively their limitation, to perform the offline analysis for every application built. At run-time, the profiler collects data from execution time and energy consumption both in the mobile device and server context. This in order to construct a cost model for the application, according to different scenarios. The offloading decision is based on the optimal solution calculated relying on both the static analysis and the dynamic profiling, and it is performed by migrating a thread from the mobile device to the clone in the server.

In 2012 Kosta et al. proposed the ThinkAir framework [42], which is a cloud-based framework that improves the idea of the MAUI and CloneCloud projects. In particular it addresses the MAUI lack of scalability using a VM-based approach and eliminates the restrictions of the offline static analysis of CloneCloud by adopting an online method-level offloading. The framework provides to the developers an API to sign which method they want to make offloadable and a specific compiler to translate the annotated code. The offloading is driven by an *Execution Controller* that takes a decision basing on the current environment data and method invocation history considering previous execution time and energy consumption. It is interesting the possibility for the user to set a policy among proposed ones. The framework provides also hardware, software and network profilers to collect various data and feed the energy estimation model used by the offloading policy.

C. Opportunistic Computing Paradigm

A first recent work that moves the attention from a centralized cloud architecture to a distributed mobile architecture is the AnyRun Computing system (ARC) [27], [28]. It explores the *opportunistic computing* paradigm [17], [16], where mobile devices are connected in an ad-hoc local wireless network to take advantage of the computing resources of other devices. The major differences from the previous solutions are that cloud or server based systems suffer from a lack of flexibility, because of the need of a specific piece of computing infrastructure known a priori. Moreover they have hidden latencies and a energy cost due to the networking communication and server power consumption. Recent year improvements of device hardware performance made possible to get over those limitations, by using mobile devices instead of servers. In this direction, ARC provides a framework to refactor the code of the application to make classes and methods offloadable to any close device, and an inference engine, based on Bayesian statistics, that decides whether and where to offload the method.
Recently the authors extend the refactoring starting from the compiled application by extracting the portable offloadable code and embedding the task code in an Android Application Package.

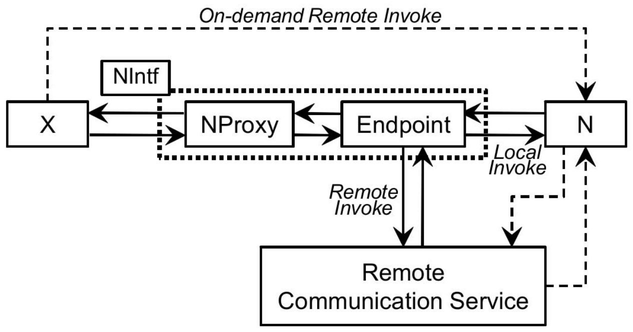

Fig. 2. DPartner: on-demand remote invocation design pattern [78].

Another recent work that exploits the *Mobile Cloud Computing* paradigm is COMPSs-Mobile [52]. It is a framework that transposes the COMPSs programming model [51] to the mobile world. This model relieves the application developers from the parallelization and distribution details. COMPSs applications are composed by annotated methods, called Core Elements (CE), that can run in parallel. The framework partitions the original Android applications during the building process, by replacing CE invocations from asynchronous tasks. These tasks are then coordinated at run-time by the toolkit basing on energy, economic and temporal cost prediction of hosting and offloading a task execution. Moreover the system comprises a check-pointing and restore mechanism to avoid a full re-execution of the application if some nodes fail.
This work marks the frontier of future offloading techniques that exploit interconnected mobile devices as a cloud system, albeit it does not exploit all the device capabilities such as GPUs or dynamic resource provisioning with a local resource manager.

D. Overall results and limitations

Due to the heterogeneous benchmarks and devices used by the different works, it is impossible to compare their results. Anyway it appears evident the benefit of using the offloading for mobile devices both for execution time and energy saving objectives, so that it will be a promising research line for the next years.

However, as stated by [59], while coexistence and deployment are reaching high results, adaptation scenarios and ease of use require further investigation. For this reason the research can reasonably be focused on an effective distributed resource management strategy, hosted on the mobile device. This should improve context-awareness and offloading policies. Certainly opportunistic offloading can get an enormous advantage from this vision because of the possibility to enrich the instances of the resource manager towards an intelligent multi-agent system, which could cooperate to achieve an energy-efficient performance speed-up.

IV. Resources Management for Mobile Systems

The aforementioned vision and the energy-budget management limitations of mobile systems suggest us to further investigate the possibility of employing a run-time resource manager on each device, in charge of performing task allocation and hardware configuration decisions. This in order to maximize an energy efficient exploitation of the distributed devices. In this work, we propose the BarbequeRTRM [5], [6], [7], which already supports embedded and distributed systems. For the last ones, MPI is currently considered the reference programming paradigm [31], [68].

The main advantages of such a vision are the possibility to manage application execution dynamically, monitoring the device status at run-time and enabling a distributed mobile computing resources allocation strategy, which will be transparent to the users. The BarbequeRTRM has been already extended in order to support Android systems and mobile oriented platform, like ARM big.LITTLE based SoC.

V. Conclusion and Future Work

In this work we presented an overview of the main approaches for distributed mobile computing and computation offloading over the last two decades. The main contributions of these works are related to the power saving and performance improvements of mobile devices, both using powerful servers and other mobile devices. Although application partitioning and distributed architectures have reached a good maturity, the resource awareness design of similar solutions needs to be investigated more deeply. We think that one of the most promising research line in the future is to integrate a run-time resource manager, such as those exposed in section IV, with the architectures and offloading techniques exposed in sections II and III, in order to create an energy and resource-aware agent that can manage local resources, schedule tasks and applications. The agents will cooperate, configuring a multi-agent system aiming at properly exploiting the distributed mobile computing infrastructure in an energy-efficient way.

References

[1] D.P. Anderson, *Volunteer computing: The ultimate cloud*, Crossroads 16, 2010.

[2] D.P. Anderson, *SETI@Home: An Experiment in Public-resource Computing*, Commun. ACM 45 ACM, 2002.

[3] M.Y. Arslan et al., *Computing while charging: Building a distributed computing infrastructure using smartphones*, Proc. 8th Int. Conf. Emerging Netw. Experiments Technol., IEEE, 2012.

[4] H. Ba et al., *Mobile computing - A green computing resource*, Proc. of IEEE WCNC Shanghai, China, 2013.

[5] P. Bellasi et al., *A RTRM proposal for multi/many-core platforms and reconfigurable applications*, ReCoSoC, 2012.

[6] P. Bellasi, G. Massari and W. Fornaciari, *Effective Runtime Resource Management Using Linux Control Groups with the BarbequeRTRM Framework*, ACM Transactions on Embedded Computing Systems (TECS), 2015.

[7] G. Massari, C. Caffarri, P. Bellasi and W. Fornaciari, *Extending a Run-time Resource Management framework to support OpenCL and Heterogeneous Systems*, PARMA-DITAM, 2014

[8] P. Bellavista, A. Corradi and C. Stefanelli, *Mobile agent middleware for mobile computing*, Computer 34, 2001.

[9] W. Binder et al., *Using Bytecode Instruction Counting as Portable CPU Consumption Metric*, Elettronic Notes in Theoretical Computer Science, 2006.

[10] BOINC, *BOINC*, BOINC's homepage [online], http://boinc.berkeley.edu.

[11] BOSP, *The Barbeque OpenSource Project*, [Online] http://bosp.dei.polimi.it/.

[12] F. Busching, S. Schildt and L. Wolf, *Droidcluster: Towards smartphone cluster computing - the streets are paved with potential computer clusters*, Proc. ICDCSW '12, 2012.

[13] E. Chen, *Offloading android applications to the cloud without customizing android*, Pervasive Computing and Communications Workshops (PERCOM Workshops), 2012 IEEE International Conference on, 2012.

[14] G. Chen et al., *Study energy trade offs in offloading computation/compilation in java-enabled mobile devices*, IEEE Trans Parallel Distrb Sys 15, 2004.

[15] B-G- Chun et al., *Clonecloud: elastic execution between mobile device and cloud*, Proceedings of the sixth conference on Computer systems, ser. EuroSys '11, 2011.

[16] M. Conti and M. Kumar *Opportunities in Opportunistic Computing*, IEEE Computer, 2010.

[17] M. Conti et al. *From opportunistic networks to opportunistic computing*, IEEE Commun. Mag. 48, 2010.

[18] L. Corral et al., *A method for characterizing energy consumption in Android smartphones*, Green and Sustainable Software (GREENS), 2013 2nd International Workshop on, IEEE, 2013.

[19] G. Coulouris et al., *Distributed Systems: Concepts and Design*, Addison-Wesley, 2011.

[20] E. Cuervo et al., *MAUI: Making Smartphones Last Longer with Code Offload*, MobiSys'10, 2010.

[21] D. Datla et al., *Wireless distributed computing: A survey of research challenges*, IEEE Commun. Mag. 50, 2012.

[22] D.Daz-Snchez et al., *Flexible Computing for personal electronic devices*, 2013 IEEE International Conference on Consumer Electronics (ICCE), IEEE, 2013.

[23] Z. Dong et al., *REPC: Reliable and efficient participatory computing for mobile devices*, IEEE int. Conf. Sensing, Communication and Networking, Singapore, 2014.

[24] J.R. Eastlack, *Extending volunteer computing to mobile devices*, Master's thesis, New Mexico State University, 2011.

[25] P.R. Elespuru, S. Shakya and S. Mishra, *MapReduce system over heterogeneous mobile devices*, Proc. 7th IFIP WG 10.2 Int. Workshop Softw. Technol. Embedded Ubiquitous Syst., 2009.

[26] H. Eom et al., *MALMOS: Machine Learning-based Mobile Offloading Scheduler with Online Training*, 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering 2015.

[27] A. Ferrari, S. Giordano and D. Puccinelli, *Reducing your local footprint with anyrun computing*, Computer Communications, 2016.

[28] A. Ferrari, D. Puccinelli and S. Giordano, *Code Mobility for On-Demand Computational Offloading*, IEEE, 2016.

[29] Folding@Home project, *Folding@Home*, [online] http://folding.stanford.edu/5.

[30] G.H. Forman and J. Zahorjan, *The challenges of mobile computing*, Computer 27. 1994.

[31] W. Fornaciari et al., *Runtime Resource Management for Embedded and HPC Systems*, Proceedings of the 7th Workshop on Parallel Programming and Run-Time Management Techniques for Many-core Architectures and the 5th Workshop on Design Tools and Architectures For Multicore Embedded Computing Platforms, 2016.

[32] C. Funai et al., *Extending volunteer computing through mobile ad hoc networking*, 2014 IEEE Global Communications Conference, IEEE, 2014.

[33] B. Goddfrey, *A Primer on Distributed Computing*, 2008.

[34] M.S. Gordon et al., *Comet: code offload by migrating execution transparently*, Proceedings of the 10th USENIX conference on Operating Systems Design and Implementation, USENIX Association, 2012.

[35] X. Gu et al., *Adaptive offloading inference for delivering applications in pervasive computing environments*, IEEE International conference on pervasive computing and communications, 2003.

[36] Y. Gu et al., *An empirical study of high availability in stream processing systems*, Middleware '09, 2009.

[37] S. Gurun and C. Krintz, *Addressing the energy crisis in mobile computing with developing power aware software*, Technical Report, Department of Computer Science, University of California, Santa Barbara, 2003.

[38] S. Gurun, C. Krintz and R. Wolski, *NWSLite: A light-weight prediction utility for mobile devices*, International conference on mobile systems, applications and services, 2004.

[39] S. Hao, *Estimating Android Applications' CPU Energy Usage via Bytecode Profiling*, Proceedings of the First International Workshop on Green and Sustainable Software, IEEE Press, 2012.

[40] P. Jain et al., *MC^2: On-the-Fly Mobile Compute Cloud for Computational Intensive Task*, ICARE, 2013.

[41] A.D. Joseph et al., *Rover: a toolkit for mobile information access*, ACM symposium on operating systems principles, 1995.

[42] S. Kosta et al., *Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading*, INFOCOM, 2012 Proceedings IEEE, 2012.

[43] D. Kotz et al., *Agent Tcl: targeting the needs of mobile computers*, IEEE Internet Computing 1, 1996.

[44] D. Kovachev, T. Yu and R. Klamma, *Adaptive computation offloading from mobile devices into the cloud*, Parallel and Distributed Processing with Applications (ISPA), IEEE 10th International Symposium on, 2012.

[45] U. Kremer, J. Hicks and J. Rehg, *A compilation framework for power and energy management on mobile computers*, Proceedings of the 14th international conference on Languages and compilers for parallel computing, 2003.

[46] K. Kumar et al., *A Survey of Computation Offloading for Mobile Systems*, Mob. Netw. Appl. Springer-Verlag New York, Inc., 2013.

[47] Z. Li, C. Wang and R. Xu, *Computation offloading to save energy on handheld devices: a partition scheme*, International conference on compilers, architecture, and synthesis for embedded systems, 2001.

[48] Z. Li, C. Wang and R. Xu, *Energy impact of secure computation on a handheld device*, IEEE international workshop on workload characterization, 2002.

[49] S. Libutti, G. Massari and W. Fornaciari, *Addressing Task Co-scheduling on Multi-core Heterogeneous Systems: An Energy-Aware Perspective*, HIPEAC Workshop on Energy Efficiency with Heterogeneous Computing (EEHCO), 2015.

[50] J. Liu et al., *Application partitioning algorithms in mobile cloud computing: Taxonomy, review and future directions*, Journal of Network and Computer Applications, 2015.

[51] F. Lordan et al., *Services: An interoperable programming framework for the cloud*, Journal of Grid Computing, vol. 12, n.1, 2014.

[52] F. Lordan and R.M. Badia, *COMPSs-Mobile: parallel programming for Mobile-Cloud Computing*, IEEE/ACM International Syumposium on Cluster, Cloud and Grid Computing, 2016.

[53] V. March et al., *μcloud: towards a new paradigm of rich mobile applications*, Procedia Computer Science, 2011.

[54] E.E. Marinelli, *Hyrax: Cloud computing on mobile devices using mapreduce*, Masters thesisCarnegie Mellon University, 2009.

[55] Y. Mustafa et al., *CWC: A Distributed Computing Infrastructure Using Smartphones*, IEEE TRANSACTIONS ON MOBILE COMPUTING, IEEE, 2015.

[56] Y. Nimmagadda et al., *Realtime moving object recognition and tracking using computation offloading*, IEEE international conferenced on intelligent robots and systems, 2010.

[57] B.D. Noble and M. Satyanarayanan, *Experience with adaptive mobile applications in Odyssey*, Mobile Netw Appl 4, 1999.

[58] K.J. O'Hara, *Autopower: toward energy-aware software systems for distributed mobile robots*, IEEE international conference on robotics and automation, 2006.

[59] G. Orsini, D. Bade and W. Lamersdorf, *Context-Aware Computation Offloading for Mobile Cloud Computing: Requirements Analysis, Survey and Design Guideline*, The 12th International Conference on Mobile Systems and Pervasive Computing (MobiSPC 2015), 2015.

[60] D.F. Parkhill, *The challenge of the computer utility*, Addison-Wesley Pub. Co. Reading, MA, 1966.

[61] C.E. Perkins, *Handling multimedia data for mobile computers*, Computer software and applications conference, 1996.

[62] B. Praveen and G. Matish, *Security Enhancement in Distributed Networking*, IJCSMC2015.

[63] P. Rong and M. Pedram, *Extending the lifetime of a network of battery-powered mobile devices by remote processing: a markovian decision-based approach*, Conference on design automation, 2003.

[64] M. Satyanarayanan et al., *The Case for VM-based Cloudlets in Mobile Computing*, IEEE Pervasive Computing, 2004.

[65] N. Seshasayee et al., *Energy aware mobile service overlays: cooperative dynamic power management in distributive systems*, International conference on automatic computing, 2007.

[66] M. Szpakowski, *Native Boinc for Android*, [online], http://nativeboinc.org.

[67] E. Tilevich and Y. Smaragdakis, *J-orchestra: automatic Java application partitioning*, European conference on object-oriented programming, 2006.

[68] R. Vavrik et al., *Precision-Aware application execution for Energy-optimization in HPC node system*, High Performance Energy Efficient Embedded Systems (HIP3ES), HiPEAC, 2015.

[69] C. Wang and Z. Li, *Parametric analysis for adaptive computation offloading*, ACM SIGPLAN conference on programming language design and implementation, 2004.

[70] J. Wernsing, *Elastic Computing: A Framework for Transparent, Portable, and Adaptive Multi-core Heterogeneous Computing*, SIGPLAN Not. 45 ACM, 2010.

[71] J.E. White, *Mobile agents*, Software agents, MIT Press, 1997.

[72] R. Wolski et al., *Using bandwidth data to make computation offloading decisions*, IEEE international symposium on parallel and distributed processing, 2008.

[73] D. Wong, *Java-based mobile agents*, Com ACM 42, 1999.

[74] D. Wong et al., *Concordia: an infrastructure for collaborating mobile agents*, International workshop on mobile agents, 1997.

[75] C. Wu and L. Li, *WIPdroid A Two-way Web Services and Real-time Communication Enabled Mobile Computing Platform for Distributed Services Computing*, Services Computing, 2008. SCC '08. IEEE International Conference on, IEEE, 2008.

[76] C. Xian, Y.H. Lu and Z. Li, *Adaptive computation offloading for energy conservation on battery-powered systems*, International conference on parallel and distributed systems, 2007.

[77] Y. Xiong, S. Huang and M. Wu, *Shared Resource and Service Management for Mobile Transparent Computing*, High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing (HPCC_EUC), 2013 IEEE 10th International Conference on, IEEE, 2013.

[78] Y. Zhang, *Refactoring Android Java Code for On-demand Computation Offloading*, ACM, 2012.

[79] Y.X. Zhang, *Transparent computing: concept, architecture and example*, Chinese Journal of Electronics, 2004.

[80] Z. Zhang, *A Hybrid approach to high availability in stream processing systems*, ICDCS '10, 2010.