UNIVERSIDAD NACIONAL DEL CENTRO DE LA PROVINCIA DE BUENOS AIRES
FACULTAD DE CIENCIAS EXACTAS

TESIS DE DOCTORADO EN CIENCIAS DE LA COMPUTACIÓN

An energy-aware scheduling approach for resource-intensive jobs using smart mobile devices as resource providers

by
Ing. Matías Eberardo Hirsch Jofré

Advisor: Ph.D Cristian Mateos
Co-advisor: Ph.D Alejandro Zunino

Tandil, Noviembre de 2017

Agradecimientos

Al Concejo Nacional de Investigaciones Científicas y Técnicas de la República Argentina (CONICET) por la beca otorgada para la realización del doctorado y a la Agencia Nacional de Promoción Científica y Tecnológica de la nación (ANPCyT) por el financiamiento otorgado al grupo de investigación en distintos proyectos. A mis orientadores, Cristian y Alejandro, por la motivación y confianza que me brindaron siempre, y por despertar mi entusiasmo por la investigación, que hoy sigue intacto, y comenzaba su camino formal aquel día en un popular café de la ciudad de Tandil. A ellos también, mis más sincero agradecimiento por la paciencia, el constante seguimiento, sus consejos y el apoyo moral que me brindaron a lo largo de todo el doctorado. A mis compañeros de doctorado, contemporáneos y los que se unieron a la cruzada más tarde. También los que transitaban esa etapa formativa hacía poco y no tan poco tiempo, sin olvidar a los compañeros de oficina. A todos ellos gracias por compartir almuerzos y charlas de los más variados temas, y quienes por el sólo hecho de estar me brindaron generosamente contención. Al grupo 5 y agregados con quienes comparto gustosos asados y cerveza artesanal elaborada por el grandote del grupo. Al grupo de amigos cuya amistad comenzó hace catorce años en la pensión de Vivi y aún se mantiene viva. Al grupo de amigos de mi ciudad natal, a quienes hace mucho que no visito pero recuerdo siempre. A mi familia toda y la familia de mi esposa, por interesarse en lo que hago. A mis hermanas por confiar en mí. A mi madre, por contagiarme su entusiasmo en lo que emprendo. A mi padre, por transmitirme su visión práctica de las cosas. A Miguel y Leticia, por compartir su alegría en todo momento.

Y a mi esposa, la luz de cada uno de mis días, la que conoce cada minuto de esta etapa importante en mi vida y a quien le estoy inmensamente agradecido por su entrega, paciencia y compresión.

Summary

The ever-growing adoption of smart mobile devices is a worldwide phenomenon that positions smartphones and tablets as primary devices for communication and Internet access. In addition to this, the computing capabilities of such devices, often underutilized by their owners, are in continuous improvement. Today, smart mobile devices have multi-core CPUs, several gigabytes of RAM, and ability to communicate through several wireless networking technologies. These facts caught the attention of researchers who have proposed to leverage smart mobile devices aggregated computing capabilities for running resource intensive software. However, such idea is conditioned by key features, named *singularities* in the context of this thesis, that characterize resource provision with smart mobile devices. These are the ability of devices to change location (*user mobility*), the shared or non-dedicated nature of resources provided (*lack of ownership*) and the limited operation time given by the finite energy source (*exhaustible resources*).

Existing proposals materializing this idea differ in the singularities combinations they target and the way they address each singularity, which make them suitable for distinct goals and resource exploitation opportunities. The latter are represented by real life situations where resources provided by groups of smart mobile devices can be exploited, which in turn are characterized by a social context and a networking support used to link and coordinate devices. The behavior of people in a given social context configure a special availability level of resources, while the underlying networking support imposes restrictions on how information flows, computational tasks are distributed and results are collected. The latter constitutes one fundamental difference of proposals mainly because each networking support –i.e., ad-hoc and infrastructure based– has its own application scenarios. Aside from the singularities addressed and the networking support utilized, the weakest point of most of the proposals is their practical applicability. The performance achieved heavily relies on the accuracy with which task information, including execution time and/or energy required for execution, is provided to feed the resource allocator.

The expanded usage of wireless communication infrastructure in public and private buildings, e.g., shoppings, work offices, university campuses and so on, constitutes a networking support that can be naturally re-utilized for leveraging smart mobile devices computational capabilities. In this context, this thesis proposal aims to contribute with an easy-to-implement scheduling approach for running CPU-bound applications on a cluster of smart mobile devices. The approach is aware of the finite nature of smart mobile devices energy, and it does not depend on tasks information to operate. By contrast, it allocates

computational resources to incoming tasks using a node ranking-based strategy. The ranking weights nodes combining static and dynamic parameters, including benchmark results, battery level, number of queued tasks, among others. This node ranking-based task assignment, or *first allocation phase*, is complemented with a *re-balancing phase* using job stealing techniques. The second allocation phase is an aid to the unbalanced load provoked as consequence of the non-dedicated nature of smart mobile devices CPU usage, i.e., the effect of the owner interaction, tasks heterogeneity, and lack of up-to-date and accurate information of remaining energy estimations. The evaluation of the scheduling approach is through an in-vitro simulation. A novel simulator which exploits energy consumption profiles of real smart mobile devices, as well as, fluctuating CPU usage built upon empirical models, derived from real users interaction data, is another major contribution. Tests that validate the simulation tool are provided and the approach is evaluated in scenarios varying the composition of nodes, tasks and nodes characteristics including different tasks arrival rates, tasks requirements and different levels of nodes resource utilization.

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

AP Access Point

API Application Programming Interface

BOINC Berkeley Open Infrastructure for Network Computing

CO2 Carbon Dioxide

DAG Directed Acyclic Graph

DES Discrete Event-based Simulation

DTN Delay Tolerant Network

DVFS Dynamic Voltage Frequency Scaling

E-SEAS Enhanced Simple Energy-Aware Scheduler

FaaS Function as a Service

FLOPS Float-point Operations per Second

FLOTES Float-point Operations To be Executed per Second

FWC Future Work aware Criterion

FVR Fixed Virtual Resource

GPS Global Positioning System

GPU Graphics Processing Unit

HPC High Performance Computing

IaaS Infrastructure as a Service

JEC Job Energy-aware Criterion

MAE Mean Absolute Error

MANET Mobile Ad-hoc Network

MCC Mobile Cloud Computing

MCT Minimum Completion Time

MCrC Mobile Crowd Computing

MFLOP Mega Float-point Operations

MFLOPS Mega Float-point Operations per Second

MSE Mean Squared Error

MVR Mobile Virtual Resource

OppNet Opportunistic Network

OS Operating System

P2P Peer to Peer

PaaS Platform as a Service

PDA Personal Digital Assistant

QoS Quality of Service

RA Resource Allocation

RSSI Received Signal Strenght Indicator

SaaS Software as a Service

SEAS Simple Energy-Aware Scheduler

SMD Smart Mobile Device

SOC State Of Charge

VANET Vehicular Ad-hoc Network

WLAN Wireless Local Area Network

Chapter 1

Introduction

1.1 Overview

Wireless communication is present in an ever increasing number of devices including sensors, wearable computers, laptops, smart-phones and tablets. Moreover, the names that have been used to refer to such devices depend on their characteristics. For instance, the term *wireless device* suggests not only devices which use the air as data transferring medium, but also those that operate with batteries as main power supply, i.e., without the need of being plugged to the electricity grid. Besides, it has been common to call wireless devices as *portable devices*, because they are small enough to be carried or worn by people. Indeed, when portable devices are intended to keep rendering service while moving, it is usual to call them *mobile devices*. Moreover, when a mobile device is capable of sensing, visualizing and performing complex computations, a frequent term used to refer to such a device is *Smart Mobile Device (SMD)*.

Today's SMDs are general purpose devices, equipped not only with several radios used to transfer data through different communication protocols –e.g., Bluetooth, WiFi, different generation of cellular networks– but also with many sensors to monitor context-aware data. They also have several Gigabytes of storage capacity and, last but not least, powerful computing resources supported by several Gigabytes of RAM, powerful Graphics Processing Unit (GPU) and multi-core processors.

SMDs capabilities and their role within the IT world have evolved over the years. From the mid 90s to the early 2000s, SMD-like devices, such as Personal Digital Assistant (PDA)s, have eased the way people access information Imielinski and Badrinath (1994); Banavar et al. (2000) and computing services Yi and Livny (1999); González-Castaño et al. (2003). A high adoption of such devices with access to resource rich infrastructures from anywhere, at anytime, is a phenomenon envisioned by Mark Weiser with the name of Ubiquitous Computing Weiser (1993), later renamed as Pervasive Computing Satyanarayanan (2001).

Nowadays, SMDs are best represented by smartphones and tablets, which are globally adopted and used as primary devices for multiple purposes, not only for providing their owner a way of communicating and accessing the Internet, but also for performing geo-localized navigation, shopping, entertainment,

and the list continues. However, an SMD owner is not the only who benefits from the capabilities of such devices. With the explosion of applications in the field of smart cities, e-government and health--care monitoring that exploit data provided by crowds of SMDs, an entire population can actually be benefited. Examples of services offered through these platforms are monitoring noise levels and online traffic information. The high popularity, low cost, and potential applications of today's SMDs contribute in that the pervasive computing phenomenon is not longer a vision but a reality.

1.2 Motivation

With the evolution of the semiconductor industry, SMDs started being equipped with faster and more computing resources, higher resolution screens and more data storage capacity. Every new SMDs generation surpasses the previous one, or even previous generations of fixed computers, in respect to the type of applications they are able to execute and the amount of data they are able to store and visualize. This fact modified the role of SMDs in the IT world, because they evolved from simple terminals to get access to the Web to small-computers with capabilities to execute resource intensive applications, e.g., face recognition, augmented reality applications Takacs et al. (2011).

Despite SMDs are powerful enough to execute such applications, a new issue entered the scene: the limited energy of SMDs that prevents applications from terminating successfully, or executing with the user desired Quality of Service (QoS) as consequence of the Operating System (OS) energy saving policies that degrade the system performance. To some extent, this issue is attributed to the development of batteries, which did not follow the growing pace of SMDs capabilities.

To allow resource-intensive applications execute in an SMD, even with resources powered by exhaustible energy sources -battery-, active research lines promote the delegation of power-hungry applications code to external fixed servers with "infinite" resources. When servers are located in a remote Cloud, the execution model is known as Mobile Cloud Computing (MCC) Abolfazli et al. (2014); Fernando et al. (2013), and the technique used to delegate code is commonly referred as cyber foraging or computation offloading Yousafzai et al. (2016); Rego et al. (2016); Sharifi et al. (2012). Among the MCC challenges in the client side, i.e., from SMDs perspective, it is the study of appropriate algorithms for partitioning applications Liu et al. (2015) to identify off-loadable code whose execution can be performed outside the SMD. Other challenges are providing decision making criteria, operating inside an SMD, to trade-off between energy saved and QoS achieved during the execution of partially-offloaded application code. In few words, the efforts of the aforementioned research lines are oriented to mitigate resource exhaustion and/or computing performance issues of an SMD while executing complex applications. Then, cyber foraging and computational offloading complement SMD execution capabilities with resources from powerful servers, i.e., SMDs are identified with the role of resource consumers.

Instead of empowering individual SMDs computing capabilities with resources from powerful infrastructures, i.e., considering them as resource consumers only, alternative research lines, with which this thesis is aligned to, advocate to consider groups of SMDs as computing resource providers. I will use the latter three terms to refer to exploitation of CPU cycles. The potential aggregate computing resources of SMDs

is non-negligible given the high adoption of such devices around the globe and the individual hardware performance -frequently underused by the owner-. Moreover, SMDs are equipped with processors that comparatively consume less energy than those of fixed computers Ba et al. (2013). All these facts mean that the development of techniques for scavenging resources in such devices would contribute not only to increase the available computing power but to reduce the emissions of Carbon Dioxide (CO2).

The computing potential of groups of SMDs has been and is currently being considered by multiple research communities which have been involved in the development of high computing power infrastructures and paradigms, namely Grid Computing Rodriguez et al. (2011); Furthmüller and Waldhorst (2010), High Performance Computing (HPC) and cluster computing Rajovic et al. (2013); Büsching et al. (2012), and volunteer computing Black and Edgar (2009). To cite a concrete example, Berkeley Open Infrastructure for Network Computing (BOINC)[1], a volunteer computing platform that exploits the computing cycles donated by millions of desktop computer users for performing scientific jobs calculation, released in June 2013 the BOINC for Android application[2]. The application allows volunteers to contribute with computing cycles of their smartphones and tablets while these are charging. The community that maintains the platform has also initiated the development of an iOS client version for the same purpose.

The inclusion of SMDs as providers of computing capability reaches sub-areas of MCC. The virtual Cloud provider Huerta-Canepa and Lee (2010), mCloud Miluzzo et al. (2012) or Hybrid Local Mobile Cloud Wei et al. (2013) are examples of such inclusion where computing resources of a group of nearby SMDs are offered as a Cloud-like execution service. Solutions like these have been proposed as an alternative for augmenting computing capabilities of an SMD without relying on private Cloud services and for comparatively reducing communication latency of tasks offloaded to remote servers.

1.3 Problem statement

As a group of SMDs essentially represents a loosely coupled distributed computing system, one concern for scavenging the aggregated computing resources relates to how SMDs are arranged, i.e., the networking support under which nodes exchange tasks and information. SMDs can be arranged via an ad-hoc network Castro et al. (2010) or, the one assumed in this thesis, an infrastructure-based network Lee et al. (2013). The network type imposes restrictions on how data flows and in turn influence resource discovery and tasks allocation.

To allocate tasks into SMDs, an energy-aware Resource Allocation (RA) or scheduling criteria is needed, which needs to be re-designed w.r.t. RA for distributed computing systems with fixed computers Zhang et al. (2016); Hussain et al. (2013); Krauter et al. (2002) to contemplate special properties of SMD. In this thesis, these properties are named SMD singularities or simply *singularities*. The term is used to refer to three unique properties of nodes that are representative of a set of issues that motivate the research in the area. The three singularities are:

[1] `https://boinc.berkeley.edu/`

[2] `https://play.google.com/store/apps/details?id=edu.berkeley.boinc&hl=en`

- *User mobility*, seen as the possibility of an SMD to change its location.
- *Lack of ownership*, seen as the non-dedicated nature of computing resources provided,
- *Exhaustible resources*, seen as the the limited operation time imposed by the finite energy supplied by batteries.

These singularities, coupled to the heterogeneous hardware capabilities of SMDs, introduce rather complex and dynamic resource availability conditions, and therefore efforts in the area proposing RA criteria can be analyzed from the perspective of the singularities they address.

In this sense, RA proposed in the literature addressing mobility in ad-hoc networks and infrastructure-networks differ in how each singularity affects resources availability. In the former, SMDs movements causing intermittent connections are treated as the normal operation scenario under which RA should be performed. Indeed in the latter case, SMDs intermittent connection is treated as a faulty situation, and nodes with low rate of connection tend to be discarded as candidates for executing tasks.

With regard to the exhaustible resources singularity, RA mechanisms targeting ad-hoc networks propose criteria focused on preserving network reachability. Indeed, this objective does not characterize RA mechanisms targeting infrastructure-based networks because the network reachability is stable and defined by the coverage range of a fixed node, which operates with energy from the electricity grid and has direct communication with all SMDs offering computing resources. The focus here, indeed, is on greedily exploiting the computing capability of SMDs.

In respect to RA addressing lack of ownership issues, as I will point out in Chapter 2 in more detail, scarce attention has been paid by the community to a topic that naturally interfere with the exploitation of computing resources. The works proposing some advance in this line model user interaction with operating and interaction data not derived from real SMDs and do not relate the effect of such interaction with the consumed energy.

Irrespective of the combination of singularities addressed by a particular RA mechanism, there are other relevant aspects to judge any solution proposed. One concerns applicability, i.e., practical implementation of the scheduling algorithm and methodology employed for evaluating it. With regard to the first, it is observed that, to operate, many state-of-the-art RA mechanisms heavily depend on accurate input task information which is very hard to obtain and generalize. To be more specific, the energy consumed by a CPU-intensive task is hard to be generalized because, not only SMDs battery depletion is not linear, but also different SMD hardware utilize the energy in a more or less efficiently manner. Many of state-of-the-art RA mechanisms require that information of energy spent by a task be available for every candidate SMD. In turn, to know the energy spent by a task, it is first necessary to know the time that the task requires on an specific SMD to execute. Knowing such time for the general case requires to first respond the question if the task execution actually terminates, which means solving the halting problem. Many state-of-the-art RA mechanisms are hard applicable because they depend on knowing the time and energy spent by tasks to operate.

As said, the other concern relates to the evaluation methodology. Given the complexity of configuring and reproducing experimental scenarios in the area, simulation is an accepted practice for evaluating RA

mechanism Hirsch et al. (2016). However, a recurrent practice when simulating SMDs clusters is to use synthetic models to represent SMDs energy depletion. These models assume that the battery depletion rate can be represented by a single linear function, which is likely to introduce serious estimation errors that compromise the accuracy of simulation results. Such models do not reflect the real SMDs energy consumption, mostly because CPU usage is never constant. This is caused by the fluctuating CPU usage caused by the owner interaction, and when it is not present, by the intervention of SMDs operating system. Other works Viswanathan et al. (2015); Mtibaa et al. (2013); Shi et al. (2012); Comito et al. (2011) which do not model energy consumption through synthetic models, utilize information profiled from the execution of real applications on real SMDs. Despite this method overcomes the aforementioned drawbacks of synthetic models, it lacks generality, i.e., simulations are valid only for a specific combination of tasks coded in one language and executed in an specific SMD.

The need of a pragmatic RA mechanism for allocating CPU-bound tasks among SMDs, that considers the resource exhaustion and lack of ownership singularities is the main motivation of this thesis. The approach combines a faithful representation of SMDs energy consumption, that includes modeling user interaction effects on tasks execution and available energy, with the easy-to-reproduce advantages of simulation.

1.4 Thesis overview and organization of the document

I propose a two-phase energy-aware scheduling approach that assigns CPU-bound tasks in a group of SMDs utilizing practical battery-aware criteria for targeting more completed tasks per energy unit than traditional (i.e., energy-unaware) heuristics. The approach is evaluated using a dual in-vitro scheme that uses energy consumption traces of real SMDs and empirical information of owner interaction load. The approach operates assuming an infrastructure-based networking support. With regard to the addressed singularities combination with which efforts are usually identified in the area, the proposal falls into the set of works classified as aware of exhaustible resources and in the set of works addressing lack of ownership from an evaluation perspective, according to the taxonomy presented in Chapter 2. The first is because for estimating the computing capabilities of an SMD, it is employed novel criteria built upon information of battery level updates reported by devices and battery capacity information provided by device manufacturer. The second is attributed to the fact that the approach evaluation includes scenarios which simulate owner interaction effects using an empirical model derived from real mobile user interactions Falaki et al. (2010). Overall, the strengths of the approach are not only the singularities combination it addresses and the in-vitro evaluation methodology employed, but also in the ease with which it can be implemented since it does not depend on hard-to-obtain information of tasks to operate, but on easy-to-obtain SMDs information.

The criteria used in the two–phase scheduling approach does not utilize tasks information to operate other than the number of queued tasks on each SMD. Tasks of varying sizes are assigned to SMDs during a *first phase* as they arrive to a centralized scheduler and the goal is to utilize the available energy efficiently so as to increase the amount of completed tasks per energy unit. During the first phase, tasks are assigned using novel energy aware criteria. As the execution of the assigned tasks to an SMD

progresses, unbalanced load is expected to emerge as consequence of heterogeneous tasks sizes –which are unknown to the scheduler– energy estimation errors and owner interaction. The unbalanced load is mitigated with a *second* allocation phase or *re-balancing phase*. In this way, the two-phase approach builds upon the hypothesis stated below:

- Hypothesis 1: Task scheduling criteria that estimate future SMDs potential computing capabilities using energy-related factors achieve better throughput than criteria that do not use it.
- Hypothesis 2: The system dynamics, inaccurate estimations of SMD computing potential and unknown tasks requirement partially break previous energy-aware task scheduling decisions creating sub-exploited slots of computing cycles that can be exploited through a dynamic task re-balancing mechanism.
  - Hypothesis 2.1: The improvement achieved by such re-balancing mechanism is conditioned by the energy-aware scheduling decisions made in the past, namely the first scheduling phase.

The content of this thesis is structured as follows. Chapter 2 presents concepts related to common networking supports adopted by the state-of-the-art RA mechanisms and deeps into the reasons behind why SMDs singularities motivate the investigation on new RA mechanisms. In line with this, the chapter also includes a section where existing RA mechanisms are classified and described according to the resource exploitation opportunity each one best fits. A resource exploitation opportunity is a concept introduced in this thesis that relates the dynamics of a context where SMDs computing capabilities aim to be exploited, with issues posed by the SMDs singularities themselves. Different resource exploitation opportunities are identified by a unique combination of singularities, and these help to coarsely identify the appropriateness of an RA mechanism for tackling issues that characterize a resource exploitation scenario.

The singularities that motivate the research in the area, and an analysis of how RA mechanisms have addressed each of them is also included in a dedicated section of Chapter 2. Such analysis includes a differentiation of the type of problems each singularity poses to resource exploitation and the methodological schemes employed in the literature for validating the existing proposals. The chapter concludes with a section that discusses the main approaches that guide existing RA solutions to address different singularities, and the practical applicability limitations and evaluation methodologies adopted by the analyzed works.

Chapter 3 describes the motivation of my proposal, delineates the problem definition and presents the hypothesis of the problem. The chapter concludes with a detailed explanation of the proposed RA approach.

Chapter 4 deeps into details of the methodology employed for evaluating the approach. The chapter describes the fundamentals of a simulation tool proposed in this thesis based on Discrete Event-based Simulation (DES) principles for Android-based mobile devices, whose implementation is open source and based on Java. The tool uses an in-vitro scheme for simulating SMDs energy depletion. It means that battery decay rate is not represented by a single linear function as other works do Singh and Raza

(2017); Vaithiya and Bhanu (2012); Ilavarasan and Manoharan (2010). Assuming this for the whole battery depletion process of an SMD is likely to introduce serious estimation errors that compromise the accuracy of the simulation process. By contrast, the proposed simulation represents battery decay rate through the composition of multiple functions that were derived through the profiling of real SMDs at discrete CPU usages. The lack of ownership singularity of SMDs, i.e., the CPU shared with owner processes and applications is supported by re-utilizing and interleaving discrete CPU usage profiles. The interleaving is performed with information derived from the study of real user interactions. Dedicated sections contained in Chapter 4 describe the algorithms and supporting applications involved in the profiling and interleaving procedures. The chapter concludes with a section that presents tests to assess the validity of the proposed simulator.

Chapter 5 delineates the experimental scenarios designed to evaluate the two-phase scheduling approach, presents the tests results and implications. Lastly, Chapter 6 delineates the final conclusions, limitations and future works.

Chapter 2

Background and related works

2.1 Preliminary concepts

Resource allocation is a central theme in distributed computing environments since it determines the effectiveness with which resources are utilized for accomplishing computational tasks (from now on simply “tasks”). Broadly, resource allocation is described as a decision-making process that allocates resources to tasks. In the literature, this general process have been referred in multiple ways, including “task mapping”, “matchmaking”, “task distribution”, “task scheduling”, “resource selection”, “resource provision”, among others. In this thesis, ”resource allocation” or “task scheduling” will be used interchangeably. Moreover, a task will be used to refer to an atomic computation whose execution starts and finishes on the same node. Besides, the terms “node” and “SMD” will be used interchangeably to refer to a mobile device resource provider.

In distributed computing environments like clusters, Grids or Clouds, resource allocation is performed over infrastructures exploiting wired links, because resources are provided by fixed machines connected through wired interfaces. However, when considering arrangements of SMDs resource allocation occurs over wireless links. Section 2.1.1 provides details of the typical forms of arranging SMDs adopted by the state-of-the-art resource allocation proposals. With regard to resources, SMDs provide context sensitive, communication and computing capabilities Pérez-Torres et al. (2016). Individually, none of these capabilities are exclusively found in SMDs. For instance, wireless communication and sensing capabilities are found in wireless sensors, and many RA mechanisms have been proposed for exploiting such capabilities, specially in the sensor networks research area Yick et al. (2008); Akyildiz et al. (2002). The same occurs with SMDs computing capabilities, since these are not different from those found in fixed hardware, for which also many RA mechanisms have been proposed Hussain et al. (2013).

Then, if SMDs resources are not distinctive and there are many works that propose RA mechanisms for exploiting such resources, an evident question is what makes the process of allocating resources in networks of SMDs different from that of allocating resources offered in networks of fixed computers. In short, the reasons are the SMDs *singularities* derived from their unique mixture of capabilities, limitations and intended purpose. A detailed explanation of these singularities is provided in Section 2.1.2.

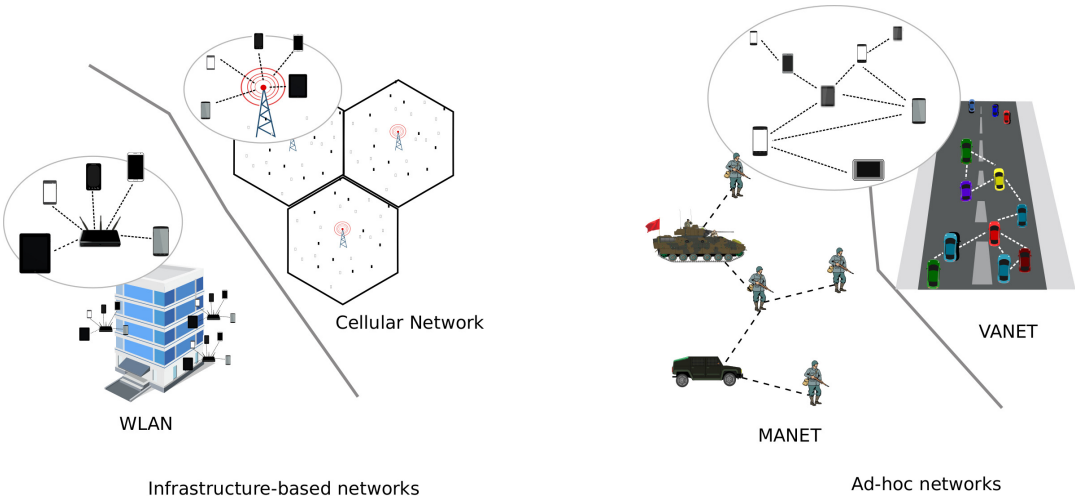


Figure 2.1: Networking supports commonly adopted for arranging SMDs

More explicitly, SMDs are not single-purpose devices like sensors but multi-purpose like PCs. However, they cannot deliver "infinite" resources like PCs because, like sensors, they rely on batteries to operate. Moreover, unlike many kinds of sensors, SMDs batteries support several charging-discharging cycles, which means that, despite their resources are not available continuously over time, these are available for several discrete time periods. Moreover, like PCs, SMDs are used by people, which means that they cannot be considered dedicated devices to purpose-specific tasks anytime, anywhere, as the case of sensors. However, like sensors, SMDs use wireless networks and the communication is feasible even while they are moving, nonetheless, such characteristic diversifies the energy costs of maintaining an active connection and transferring data. Therefore, the dynamic operation and heterogeneity introduced by the above features pose new challenges to the resource allocation process when resources to be exploited are provided by SMDs.

2.1.1 Networking support for coordinating SMDs collaboration

In order to exploit SMDs resources an RA mechanism must rely on a networking support to perform activities such as acquiring knowledge of resource availability, distributing tasks and collecting results. Figure 2.1 illustrates the two most common alternatives for arranging SMDs, infrastructure-based and ad-hoc networks. Table 2.1 complements Figure 2.1 with research works targeting each type of network support. Either alternatives for arranging SMDs presents advantages, disadvantages and open challenges that are subject of study within the communication research area. As they are out of the scope of this thesis, I will not provide insights on these challenges other than an overview of key features that affect the resource allocation process.

Infrastructure-based networks are characterized by the existence of dedicated hardware, known as base stations, which operate with energy from the electricity grid and render wireless communication services within the range of the signal coverage. Nodes willing to communicate with other nodes do it through the services provided by the base station. Examples of base stations are wireless access points and cellular towers. A proxy-based setting relies on such type of networking support to abstract details of SMDs

Networking support	RAproposals
Ad-hoc	Shi et al. (2016); Ghasemi-Falavarjani et al. (2015); Shah (2015); Mtibaa et al. (2013); Shah et al. (2012); Shi et al. (2012); Furthmüller and Waldhorst (2012); Murray et al. (2010); Shivle et al. (2006)
Infrastructure-based	Rodriguez et al. (2014); Lee et al. (2014); Birje et al. (2014); Chunlin and Layuan (2014); Wei et al. (2013); Vaithiya and Bhanu (2012); Jang and Lee (2011); Ghosh and Das (2010); Ilavarasan and Manoharan (2010); Park et al. (2003)

Table 2.1: RA mechanisms differentiated by networking support

resources to higher levels of a hierarchical resource organization that can be connected through wired links. The knowledge of resources availability within the coverage range of a base station is managed by a special fixed node called proxy where the scheduling logic resides.

The other networking support for arranging SMDs is via an ad-hoc multi-hop network that shares similarities with Peer to Peer (P2P) networks with respect to self-organization, decentralized control capabilities, and connectivity in highly dynamic environments. Examples of these type of networks are Delay Tolerant Network (DTN)s, Opportunistic Network (OppNet)s and Mobile Ad-hoc Network (MANET), where nodes operate both as hosts and routers, i.e., they contribute with resources not only for executing tasks but also for forwarding packets towards other nodes that might not be within the direct transmission range Castro et al. (2010) of the source node. Here, packet routing is a very complex task due to the changes in communication paths caused by nodes mobility, and it is also an energy-consuming task due to the complex treatment of collision and interference situations. Ad-hoc networks have been proposed to support communication where a fixed networking infrastructure is not available, e.g., in rescue operations after natural disasters Macone et al. (2013); Monares et al. (2011); Aldunate et al. (2006). The scheduling logic in such type of networking support is performed by all or selected nodes in the network.

2.1.2 SMDs singularities

Accurate resource quantification is a crucial feature for an RA mechanism to be effective. In fact, when resources are very heterogeneous and their availability dynamically varies, achieving high effectiveness becomes complex. Resources provided by SMDs are precisely very heterogeneous, which results not only from the combination of different hardware models of CPU, GPUs, sensors and wireless radios that manufacturers include in a single device, but also from the modes in which some of those circuits are able to operate Marinescu et al. (2003). For instance, CPUs can balance performance and energy consumption through Dynamic Voltage Frequency Scaling (DVFS) techniques.

Besides, another fact that contributes to resource heterogeneity is the finite energy of SMDs. Dedicated devices with the same hardware but different remaining energy are not expected to provide the same amount of resources to execute tasks. Supposing that the amount of resources only depends on the SMD remaining energy, which in turn is difficult to estimate accurately, then a rank of resource quantity could

be relatively easy to obtain by ordering SMDs by their remaining energy. However, when the remaining energy is not the only different feature of SMDs, i.e., their hardware characteristics also differ, such resources quantification becomes more complex. This complexity increases even more when considering that resources availability dynamically changes as consequence of the multipurpose and non-dedicated nature of SMDs. Below, we delineate the aspects that contribute to the resource heterogeneity and dynamic availability –from now on called SMDs singularities–, which state-of-the-art resource allocation mechanisms to some extent address for the effective exploitation of SMDs:

- *User Mobility*: It refers to the fact that SMDs location depends on their owner's location. It is a singularity that strongly affects the quantification of SMDs communication resources. The higher delays, error rates and frequent spurious disconnections that wireless communication of SMDs presents when compared to wired communication is, in part, due to user mobility. In ad-hoc networks, mobility can cause the path some data should traverse until reaching the destination node to change during the transmission process and that, in turn, modifies the time, energy and intermediate nodes involved in such end-to-end communication Rezende et al. (2011). In infrastructure-based networks SMDs mobility can increase hand-off events, which are an important source of delay in the data transmission process Liu et al. (2008). Even when path changes or hand off events do not occur, the signal quality of an SMD link is affected by user mobility and that, in turn, modifies the rate of useful data that is being transferred. User mobility may derive in connection-disconnection patterns with regard to, e.g., access points in infrastructure-based networks or other SMDs in ad-hoc networks. In certain types of networks, like DTNs or OppNets, the connection-disconnection patterns, rather than being considered faulty situations, are treated as normal operation conditions under which resources should be exploited Spano and Ricciato (2016). User mobility does not only contribute as a source of heterogeneity when exploiting communication resources, but also when exploiting sensing capabilities of SMDs. As was pointed out in a discussion of a participatory sensing survey Khan et al. (2013), the uncontrolled mobility of people is challenging because it leads to the *sensor availability problem*, i.e., the rendezvous among the sensors and the communication infrastructure may not happen on the time scales best suited to the needs of applications.

- *Lack of ownership*: It refers to the non-dedicated nature of SMDs. The system or application that aims at scavenging SMDs idle resources does not have the full control of those resources. In other words, resources such as CPU time are shared with SMD owner processes and applications. Irrespective of the incentives that encourage owners to offer their SMDs' resources Restuccia et al. (2016); Duan et al. (2014); Vega et al. (2013), external tasks are expected not to (heavily) degrade the performance of owner's applications or experience. Such scavenging principle suggests that the actual resources exploitation level and availability differ from the maximum level that the SMDs hardware is able to deliver. This is not exclusive to SMDs but an issue also present in desktop PCs of volunteer computing projects, e.g., the BOINC and XtremeWeb platforms Urbah et al. (2009). However, with SMDs the non-dedicated issue is brought to foreground because SMDs are primary computing devices with which people interact very frequently during the day. For this reason, resource scavenging should be performed in such a way that it does not degrade owner's

applications performance to unacceptable QoS levels. Besides, careful energy management must be ensured, i.e., foreseeing future owner interactions.

- *Exhaustible Resources*: It is related to the fact that resources provided by SMDs are limited by, or constrained to, the energy availability of their batteries. This poses another distinctive challenge to RA mechanisms since finite energy is a new dimension of heterogeneity at the moment of quantifying the resources that an SMD is able to provide. Resource quantification is a top-priority issue of RA mechanisms Furthmüller and Waldhorst (2010); Litke et al. (2004) that target battery-driven devices. Even when resource exploitation is planned to occur while SMDs are plugged to the electricity grid, the scenario is not challenge-free. As shown in Arslan et al. (2012), such type of resource scavenging should be regulated so as to not heavily affect the time that the charging cycle lasts.

All in all, the above singularities pose new challenges to legacy RA mechanisms when using these latter to exploit resources provided by arrangements of SMDs. However, from a practical point of view, the impact of singularities on achieving effective exploitation may vary from one *resource exploitation opportunity* to another. In short, a resource exploitation opportunity involves aspects such as a place, a social context, a timescale, a type of communication support and tasks whose completion is performed with a set of SMDs. All these aspects suppose the existence of a wide variety of resource exploitation opportunities where some of the singularities may not be a concern that needs to be addressed by the RA mechanism. An example is user mobility when the SMDs involved are exploited at the same place during the same time period, like during office hours.

Indeed, the fact that most of the current RA proposals targeting SMDs as resource providers do not simultaneously address all these singularities, does not invalidate their contributions. By contrast, addressing a subset of SMDs singularities relates to the resource exploitation opportunity these proposals target. For this reason, and to avoid focusing simply on whether a work address or not a singularity, this thesis also contributes with a practical categorization of resource exploitation opportunities based on the importance or weight each singularity has w.r.t. the resource exploitation approach proposed. The categorization serves as a criterion for organizing current and future RA mechanisms targeting SMDs, as well as to better appreciate their contributions and facilitate future comparisons. Section 2.2 deepens into the idea of *resource exploitation opportunity*.

2.2 Resource exploitation opportunities

SMDs have become the primary computing device carried by people most of the time. This phenomenon attracted the attention of many researchers who see in social contexts such as libraries Loke et al. (2015), university campuses Katsaros and Polyzos (2007), conference rooms Murray et al. (2010), museums Huerta-Canepa and Lee (2010), or even outdoor public places Satyanarayanan (2010) natural opportunities for scavenging the aggregated computing capabilities. There are also researchers Mtibaa et al. (2013); Park et al. (2003) who differentiate the computing potential of groups of co-located SMDs based on their connection stability, identifying stable, unstable and very unstable arrangements, or equivalent

coarse-grained categories to described them. Other works Habak et al. (2015); Shi et al. (2012) incorporate the notion of connection predictability. In Habak et al. (2015), a stability spectrum is illustrated with real life examples. At the extreme of the least stable and predictable settings are OppNets, such as those in which works including Shi et al. (2012); Murray et al. (2010) are based on. At the other extreme, there is a stable setting exemplified with the Mont-Blanc project Rajovic et al. (2013) where a set of mobile CPUs were mounted in single chassis to exploit their energy efficient computing capabilities. In the middle of the spectrum are settings where people congregate with its personal devices at known time schedules, e.g., a classroom, a theater play, among others.

The computing potentials characterization of SMD clusters based on their connection stability covers only data transfer-related capabilities of SMDs, and leaves outside other aspects concerning their processing-related capabilities. Given that tasks scheduled to an SMD share resources with that of its owner, his/her interaction should be considered for rating the real capabilities. This involves categorizing the computing capabilities by looking at, for instance, the most used applications, frequency and length of sessions. An SMD cluster involving owners with a high-demand usage profiles may not represent the same computing potential than owners with low-demand usage profiles. The first could be the case of people entertaining themselves with a resource-intensive game while in a waiting room. The case of low-demand user profiles may refer to a group of students attending motivating lessons in classrooms of an university campus. In the middle of interaction categorization, there are clients at a coffee shop, surfing the web or reading an electronic newspaper.

The complexity that SMDs singularities suppose for RA mechanisms should not be analyzed only in terms of social contexts dynamics, but also through the networking support used to coordinate resources. To illustrate the idea, let us assume we use SMDs resources by exploiting the regularity of people driving their particular cars through a highway. Besides, consider that resources coordination is performed either through a set of cellular towers placed along the highway (infrastructure-based support) or through Vehicular Ad-hoc Network (VANET)s. With the first coordination support, by considering SMDs location information, long data transfers can be accelerated by avoiding SMDs that traverse high communication latency areas, e.g., caused by handover operations Zola and Kassler (2016); Ferretti et al. (2016). With the second coordination support, the fact that clusters of SMDs move in the same direction at a similar speed do not negatively affect the established links, and the treatments of mobility issues are designed for special cases rather than for normal operation modes.

Then, to acknowledge how complex would be to provide a holistic categorization for measuring the computing potential of SMD clusters, we do not have to forget the characteristics of the tasks Shah (2015) that SMD cluster will execute. In concrete, independent CPU-intensive tasks (e.g., number crunching) might be less affected by disconnections caused by user mobility than CPU-intensive tasks with data dependencies. While in the first case interruptions in the communication delay the collection of partial results, in the second case the entire application progress can be delayed because tasks might not be able to start until the data produced by all preceding tasks is available. Similarly, tasks with hard deadlines will be more affected by high-demand usage profiles than tasks with soft deadlines.

Identifying which singularity/ies need to be addressed on every possible resource exploitation opportunity would be impracticable due to the innumerable possibilities of combining SMDs usage contexts, net-

	$\sim LO$		LO		
	Ad-hoc	Infrastructure-based	Ad-hoc	Infrastructure-based	
ER	Shi et al. (2016); Viswanathan et al. (2015); Mtibaa et al. (2013); Shi et al. (2012); Furthmüller and Waldhorst (2012)	Singh and Raza (2017); Chunlin and Layuan (2014); Vaithiya and Bhanu (2012)	×	×	UM
$\sim ER$	N/A	N/A	×	×	$\sim UM$
$\sim ER$	Shah (2015); Loke et al. (2015); Li et al. (2015); Shah et al. (2012)	Habak et al. (2015); Lee et al. (2014)	×	Ghosh and Das (2010)	UM
ER	Ghasemi-Falavarjani et al. (2015); Comito et al. (2011)	Birje et al. (2014); Wei et al. (2013); Ilavarasan and Manoharan (2010)	×	Rodriguez et al. (2014)	$\sim UM$

Table 2.2: Map of RA works by resource exploitation opportunities resulting from a binary valuation of addressing SMDs singularities

working support and task characteristics. However, resource exploitation opportunities could be coarsely "clustered" based on the singularities combination that the RA mechanism needs to address for the effective exploitation of resources. Table 2.2 depicts such a clustering where the need for singularity support is represented as a binary value. UM refers to user mobility, LO to lack of ownership and ER to exhaustible resources. The cells of the Table shows the RA mechanisms which address the singularities of the resource exploitation opportunity cluster. Each cluster is further divided into ad-hoc and infrastructure to give an overview of the distribution of state-of-the-art RA mechanisms assuming each type of networking support. Notice that the resource exploitation opportunity where none support for singularities is needed, is marked with N/A. It indicates that outfitting RA mechanisms with special logic to cope with SMDs singularities would not be necessary for effectively exploiting resources.

An RA proposal is considered to be appropriate for the resource exploitation opportunity provided it addresses the singularities combination. Addressing a singularity suggests that the RA mechanism proposes an algorithm or a formula using specific information to cope with the issues posed by the singularity. Examples of this include the application of conditional probabilities and Markov chains to model connection/disconnection of SMDs within a zone, the proposal of SMDs rankings that combine information such as GPS location, battery charge level of SMDs to determine their suitability for executing tasks, among others. Another way of addressing a singularity is from the evaluation methodology. For instance, there are proposals that do not prescribe concrete actions to deal with user mobility, but the

evaluation includes SMDs which periodically change their location by following some defined mobility model.

Moreover, the cross symbol (×) indicates that there is no work in the literature, up to the moment of writing this thesis, which addresses the singularities combination. The next subsections provide details of the RA mechanisms that address each singularities combination. It is worth noting that the descriptions provided for the works discussed do not aim to be full characterizations of the efforts but only highlight the techniques, algorithms, and optionally type of applications targeted, that are related to the singularities combination these works address. Common features of the surveyed works complementing these descriptions can be found in the analysis presented in 2.3.

2.2.1 RA proposals that address UM_ER singularities combination

In (Shi et al., 2012), the authors provide concrete actions for dealing with user mobility and resource exhaustion by means of Serendipity, a system for disseminating tasks in a group of SMDs intermittently connected through ad-hoc links. They propose three tasks dissemination strategies based on the predictability of future contacts and the existence of a control channel for coordination. When future contacts are predictable and there is a control channel (ideal case), it is proposed a Water-filling greedy strategy that iteratively chooses the destination SMD for every task that pursues global minimization of tasks completion time. When future contacts are predictable but there is not a control channel for coordination, a Computing on Dissemination (CoD) strategy is used to distribute tasks opportunistically and tasks time minimization is performed locally. For the case with least context information, i.e., unpredictable future contacts and no control channel, the authors proposed upCoD, a variation of the CoD strategy. Instantiation of the strategies are also proposed for increasing the lifetime of participant SMDs where the residual energy is considered for tasks dissemination. Tasks are assumed to be provided with information that allow the Serendipity system to determine the execution time and energy spent in each candidate SMD. The evaluation includes a set of experiments with a simulation software using real traces and another set with a prototype implementation of Serendipity that uses real SMDs, a speech-to-text application and a face detection application.

In (Mtibaa et al., 2013), Abderrahmen Mtibaa et. al propose a resource sharing algorithm for ad-hoc networks where nodes collaborate on the execution of independent computational tasks is proposed. The objective of the algorithm is to increase network lifetime, i.e., prolong the time of the first node that fails due to a battery depletion event. The algorithm considers finite energy of nodes and intermittent connection when offloading tasks among nodes at one hop, and two-hop distance from the node that initiates the offloading process. For each task to be executed, the algorithm evaluates the energetic convenience of delegating it to a neighbor node versus executing the task locally. The criterion to define such convenience uses the communication and computation requirements of tasks, the energy spent by nodes while executing and transferring the task and the remaining energy of nodes. For dealing with intermittent connections and before evaluating the energetic convenience, a check for the existence of communication paths is performed. The paths should exist to allow the tasks distribution and results collection to happen within the tasks deadlines. The authors assume that the group of collaborating

nodes can be derived from the analysis of contact duration and frequency information. Once such group is identified, the algorithm continues with the evaluation of the energetic convenience. The proposal is evaluated with experiments that include real mobility traces and energy consumption information derived from real mobile devices.

In (Chunlin and Layuan, 2014) the authors propose a service composition mechanism for negotiating SMDs resources using agents. The focus of the proposal is on an economic model that rules the requests and offers of resources, which aims at maximizing the utility of a resource allocation by the application of a Lagrange multiplier-based approach. The resource exhaustion is contemplated by the constraint of SMDs finite capacity in the utility maximization problem statement. The user mobility singularity is not considered in the approach other than through its evaluation. The authors include simulation scenarios where SMDs move with a random-walking mobility pattern with speeds varying between 1 and 20 meters per second.

In (Furthmüller and Waldhorst, 2012), a framework for sharing resources of SMDs is proposed. The usage of a resource or group of resources are offered and consumed by applications as services. A study of several service selection criteria is presented with focus on determining the benefits on extending the battery life of service providers. The proposed selection criteria use information of service energy consumption and the remaining charge of devices. Additionally, a framework to derive the energy consumption model of SMDs is proposed. Moreover, user mobility is addressed in the same way as Chunlin and Layuan (2014), i.e., only through the evaluation of the approach.

In (Vaithiya and Bhanu, 2012), the authors propose and simulate the performance of an algorithm that predicts SMDs availability by using a physical availability factor (PAF) and a battery availability factor (BAF) for scheduling heterogeneous tasks. The authors assumed that SMDs providing resources are connected to a base station that is part of the infrastructure of a cellular network. Six adjacent cells of that network are considered a zone. The resource allocation is structured in a two-level hierarchy where the first level operates from the central cell of a zone and hides resources heterogeneity details to the inter-zone level. The PAF is used to infer SMDs future location and is determined based on its movement type, i.e., moving towards and moving apart from the base station, and the mobility pattern. The movement type is predicted based on Markov chains, SMDs Global Positioning System (GPS) information and the Haversine formula. The latter is used to calculate the distance, in this case between an SMD and a reference base station, taking into account the curvature of the earth. Moreover, the BAF value is determined with parameters such as SMDs battery capacity, C-rate, battery power usage and total battery availability. Authors propose to use both factors as criterion for measuring nodes availability.

In (Shi et al., 2016) the authors propose an RA mechanism for local mobile clouds that assigns tasks to energy efficient processing nodes with an adaptive probabilistic approach. The proposal is not aware of exhaustible resources, but the singularity is contemplated in the evaluation when authors state that SMDs are initialized with the same battery level. The RA mechanism operates as follows: for every task, it selects a set of candidate SMDs able to execute the task within a time constraint determined with information of SMDs processing capabilities. The definitive SMD in charge of the task execution is probabilistically selected from the set of candidates where probability is defined based on the energy each SMD spent on executing the task. Time constraint is, in part, defined by the task deadline, which

is provided by the task owner, minus a margin value that is calibrated when tasks fail. Task failures are detected when tasks cannot be completed within the owner task deadline and one cause of these failures is attributed to unpredictable SMDs queuing delays which are in turn consequence of assignments from multiple source nodes. The task completion rate achieved by the RA mechanism is evaluated in a simulated environment with stationary and in-movement SMDs settings, varying the number of SMDs, their computing capabilities, mobility patterns, tasks requirements, and the interval of topology control messages.

In Singh and Raza (2017) the authors present a framework that combine concepts of quantum computing and binary gravitational search to process jobs with execution dependencies. The resulting meta-heuristic (QBGSA) aims at minimizing the turnaround time of jobs. The exhaustible resources and user mobility singularities are considered from the fitness function used to evaluate the solutions quality found in each iteration of the of the evolutionary process. Precisely, the fitness function combines a BP (battery power) value and a MS (mobility score) value. The first is derived from the remaining energy percentage reported by SMDs, and the second is calculated using information of the mobility history of SMDs through different coverage ranges during a week for office time hours. The performance of QBGSA is compared with that of QGA (quantum genetic algorithm).

In Viswanathan et al. (2015), a resource provisioning framework with autonomic capabilities is proposed for scheduling independent tasks of real-time in-the-field health care applications. The devices are assigned with roles that relate to their capabilities for computing, sensing and transferring data. Nodes with the service requester (SR) role ask for some data to be processed. Nodes playing the service provider (SP) role sensor data from the environment and/or offer computing capabilities while a node with the role of arbitrator (broker) provides resource discovery and allocation services. The RA logic leverages long-term statistics as a way of managing uncertainty caused by users mobility. Statistical indicators considered are the average arrival rate of SP nodes and the average times SPs are connected and disconnected from an arbitrator. Besides, the novel concept of *applications waypoints* outfits the RA logic. It is an aid to mitigate the uncertainty of tasks execution performance caused by unpredictable SPs resource utilization, inaccurate energy estimation and tasks completion time. The RA can be tuned with policies to achieve a minimization of the maximum battery drain -lifetime maximization of the network-, or minimization of the applications response time without considering battery drain QoS maximization-. The policies operate with information of nodes availability periods as well as performance metrics such as energy consumption and resources utilization. Provider nodes voluntarily inform such information to arbitrator nodes.

2.2.2 RA proposals that address LO_UM singularities combination

In (Ghosh and Das, 2010), the authors provide some insights on the impact of user mobility and lack of ownership singularities in the completion of CPU-intensive tasks. The work is focused on proposing an economic model for resource allocation based on non-cooperative bargaining game theory. In addition to that, they assess the impact of reducing the cost of location update (mobility tracking) of SMDs in an IEEE 802.11 mobile Grid architecture by proposing a location management framework based on the

Lempel-Ziv (L778) compression algorithm. They show that using the data provided by such framework, the task allocator component, accommodates more tasks than when such data is not considered. Besides, the work contemplates the non-dedicated nature of SMDs by considering non-grid tasks competing for resources. The execution model of devices is assumed to be a M/G/1 preemptive priority queue where internal tasks arrival rate can preempt grid tasks.

2.2.3 RA proposals that address ER_LO singularities combination

In (Rodriguez et al., 2014), the authors boost the Simple Energy-Aware Scheduler (SEAS) mobile Grid scheduler presented in (Rodriguez et al., 2010) with different job stealing techniques. The way SEAS consider the exhaustible resources of nodes is by predicting remaining node uptime from battery drop events. Remaining uptime information combined with Float-point Operations per Second (FLOPS) and assigned tasks is used to rank SMDs and decide the most appropriate one for executing a newly arrived CPU-bound task. The job stealing techniques are proposed as a protection mechanism against unbalanced load produced as consequence of inaccurate remaining uptime predictions that, in turn, derive from factors such as workload, network usage and inaccurate battery sensor of SMDs. Experiments include settings with dedicated and non-dedicated SMDs. In non-dedicated settings, SMDs CPU is configured to have 30% utilization simulating an average owner CPU utilization.

2.2.4 RA proposals that address UM singularity

In (Shah et al., 2012), a Two-Phase Resource Allocation scheme (TPRA) for dealing with the user mobility singularity is proposed. TPRA exploits movements history information of SMDs and wireless communication energy consumption properties to distribute dependent tasks of varying requirements. The first phase aims at reducing the probability of communication interruptions due to SMDs mobility. When an SMD needs computing resources, it broadcasts its next probable location and SMDs willing to provide such resources and share the same location respond to the request. The next probable location is determined by means of a Markov chain that stores the history of user mobility patterns. The physical area where nodes movement is mapped to a virtual grid equal-sized cells. The states of the Markov chain are the cells while transitions from one state to other represent the movements of an SMD between cells. Movements of each SMD between cells is represented as a probability matrix whose values are updated every time an SMD moves. The second phase of the RA scheme uses physical distance information among SMDs to minimize communication latency of tasks that need to exchange data. In (Shah, 2015), an extension of Shah et al. (2012) called ERRA is proposed, which aims at minimizing SMDs energy consumption while executing a group of tasks. In ERRA's second phase, data dependent tasks are assigned to the heaviest weight k-devices group, where k is equal to the number of tasks and the weight of groups is determined with information of the SMDs transmission power level.

In (Lee et al., 2014) the authors propose a fault avoidance approach for scheduling tasks by taking into account dynamic properties related to communication capabilities of SMDs. Such properties are: availability, reliability and maintainability -whose individual values are aggregated into another single

property called effectiveness-, and d-effectiveness -that quantifies user's movements pattern-. Availability is defined as the probability that a device is operational and/or able to return results which, in turn, depends on the time period the device is able to provide its resources. Reliability is the probability that a device performs a task for a given time period without failures and maintainability is how quickly a device recovers from a failure. The product of the these properties values results in an *effectiveness value* which is used to rank devices. Device A with higher effectiveness value than device B is supposed to be available for a longer period to perform tasks, to operate with less failure probability and to recover quicker when a failure occurs than device B. D-effectiveness property is the ratio of effectiveness to the Euclidean distance between the availability, reliability and maintainability values in the present with regard to the same value measured in the past. The prediction quality of d-effectiveness depends on the fact that users exhibit regular connection patterns w.r.t to the same time-of-day and day-of-week. The evaluation includes the usage of real-life traces from SMDs.

In Habak et al. (2015), Habak et al. propose FemtoCloud, a system for scheduling independent hybrid tasks, i.e., with data and CPU cycles requirements. A prototype of the system was implemented for Android-powered SMDs. The proposal aims at exploiting the computing capabilities of a group of SMDs putting special attention in the connection periods of SMDs to the *controller*, a component in charge of the task assignments. The time at which tasks should be assigned and results collected are scheduled by two complementary greedy heuristics that take into account the tasks deadline constraints. One heuristic schedules the time and resource utilization for transferring inputs and executing the tasks. Another heuristic schedules the appropriate time for SMDs to send the tasks results. The objective is to achieve the highest amount of useful computation done. The heuristics efficiency is evaluated through simulations and real arrangements of SMDs.

2.2.5 RA proposals that address ER singularity

In (Birje et al., 2014), a variant of the economic model proposed in Ghosh and Das (2010) is presented. It does not consider user mobility related issues, neither SMDs lack of ownership but contemplates resource exhaustion when distributing tasks, a singularity that is not addressed in Ghosh and Das (2010). In the process of resource negotiation, a resource brokering agency that centralizes the information of SMDs, assigns them a *reliability score*. The score is calculated combining information of tasks requirements and SMDs resources capabilities including processing rate, memory, bandwidth and battery power. The experiments include a comparison of the proposed RA mechanism against others of the same authors (Ghosh et al. Ghosh and Das (2010)).

In (Ghasemi-Falavarjani et al., 2015), it is proposed a multi-objective two-step resource allocation approach that deals with the resource exhaustion of Ghosh and Das (2010)s. The first step employs a Non-dominated Sorting Genetic Algorithm II (NSGA-II) to obtain a set of solutions near to the Pareto-optimal front. The second step employs the entropy weight and Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) to select the best balanced solution that minimizes tasks completion time and consumed energy. Other constraints considered in the proposal are tasks user acceptance deadlines, nodes residual energy and budget. The budget constraint corresponds to a virtual payment for

offered resources that is proposed as an incentive mechanism for encouraging SMDs collaboration. The proposal is evaluated with OMMC, a context-aware offloading middle-ware, that was installed on real SMDs. The experimental scenarios include different combination of SMDs, heterogeneous tasks and comparisons Shi et al. (2012) proposal.

In (Ilavarasan and Manoharan, 2010), an RA mechanism, named HPSM, targets dependent tasks and the objective function used to guide node selection, can be tuned to prioritize the conservation of SMDs energy, to minimize tasks schedule length or to achieve a balance between both opposite goals. As HPSM is designed for dependent tasks, HPSM considers the execution capability of SMDs and communication bandwidth between them, as well as, energy spent during task execution and results transferring. The RA mechanism has a compile-time phase where tasks dependencies, represented as a Directed Acyclic Graph (DAG), are analyzed to determine groups of tasks at each level of the DAG that can be executed in parallel. Then, tasks are assigned with a priority based on amount of input data they required, the output data they generate, the sum of both amounts of data and the average computation time of the task on every candidate SMD. After that, in a node selection stage, by iterating the list of prioritized tasks, the SMD that minimizes the configured objective function is assigned with the current task. The cycle is repeated until all tasks are assigned. A component of the objective function contemplates the exhaustible resources singularity of SMDs. The performance is evaluated through simulation and comparisons against two algorithms named HEFTM and PETSM that are adapted from HEFT Topcuoglu et al. (2002) and PETS Ilavarasan and Thambidurai (2007) algorithms inherited from traditional heterogeneous computing environments, varying the number of DAG instances generated, number of SMDs and energy consumption rates of SMDs.

In (Comito et al., 2011), an RA mechanism for prolonging network lifetime in a collaborative ad-hoc network composed by SMDs is proposed. The tasks are generated by SMDs that also participate in the execution of other tasks. The RA decisions are decentralized and hierarchically organized in clusters. It means that, a task assignment is firstly evaluated among the SMDs within the scope of the local cluster where the task execution request originates. If none of the SMDs of the local cluster is able to execute it without depleting its battery, the assignment is delegated to an inter-cluster level where SMDs of other clusters are considered. To compute the feasibility of an assignment, the RA mechanism considers the energy required to compute and transfer the involved task. The evaluation is performed with a system prototype composed by real SMDs and Android emulators.

2.3 Analysis of how SMDs singularities are addressed

This section analyzes and classifies how SMDs singularities are addressed by the surveyed RA mechanisms. The analysis and classification emerges from the works that have been described in Section 2.2, and additional ones whose detailed description have been omitted due to their similarity with such works w.r.t how they address the singularities, their RA goals and the assumptions they make.

Table 2.3 gives a big picture of the amount of RA proposals that are in line with the SMD singularities explained in Section 2.1.2. One reason that makes an RA proposal to be considered in line with an SMD

Singularity	Aware		Evaluated	
	Ad-hoc	Infrastructure-based	Ad-hoc	Infrastructure-based
User mobility	Viswanathan et al. (2015); Shah (2015); Shi et al. (2012); Mtibaa et al. (2013)	Singh and Raza (2017); Habak et al. (2015); Lee et al. (2014); Vaithiya and Bhanu (2012); Jang and Lee (2011); Park et al. (2003)	Shi et al. (2016); Li et al. (2015); Loke et al. (2015); Furthmüller and Waldhorst (2012); Murray et al. (2010)	Chunlin and Layuan (2014)
Lack of ownership	×	Ghosh and Das (2010)	×	Rodriguez et al. (2014), **This thesis**
Exhaustible resources	Viswanathan et al. (2015); Ghasemi-Falavarjani et al. (2015); Shah (2015); Mtibaa et al. (2013); Shi et al. (2012); Comito et al. (2011); Furthmüller and Waldhorst (2012)	Singh and Raza (2017); Chunlin and Layuan (2014); Birje et al. (2014); Rodriguez et al. (2014); Wei et al. (2013); Vaithiya and Bhanu (2012); Rodriguez et al. (2010); Ilavarasan and Manoharan (2010), **This thesis**	Loke et al. (2015)	×

Table 2.3: SMDs singularities addressed by state-of-the-art RA mechanisms

Approach	Perception of user movement	Focus	RA works
Aware	Beneficial (it is an opportunity to reduce tasks completion time)	Search for routing paths through SMDs or potential clusters which favor tasks distribution and results collection	Shah (2015); Mtibaa et al. (2013); Shi et al. (2012)
	Harmful (it is the main cause of task failure)	Search for SMDs with good connectivity metrics in respect to the resource allocator coverage region	Singh and Raza (2017); Viswanathan et al. (2015); Habak et al. (2015); Lee et al. (2014); Vaithiya and Bhanu (2012); Jang and Lee (2011); Ghosh and Das (2010); Park et al. (2003)
Evaluated		Add realism to the evaluation of the proposal	Shi et al. (2016); Li et al. (2015); Chunlin and Layuan (2014); Furthmüller and Waldhorst (2012); Murray et al. (2010)

Table 2.4: Identified approaches for addressing user mobility

singularity is that it utilizes information in the RA logic to deal with challenges posed by the singularity, e.g., the current energy level of a device is frequently used to deal with resource exhaustion singularity. The other aspect that makes an RA proposal to consider a singularity is when it represents/includes the singularity within the experimental variables of the performance evaluation. In the first case, the RA proposal is *aware* of a singularity, while in the second case, it is *evaluated* considering a singularity. Furthermore, works that are aware of a singularity are also evaluated w.r.t. that singularity, but the opposite does not necessarily hold.

2.3.1 User mobility

As shown in Table 2.4, works that are aware of user mobility are, in turn, differentiated in the way they perceive the singularity. The fact that works such as Shah (2015); Mtibaa et al. (2013); Shi et al. (2012) perceive user mobility as beneficial, means that user movements are exploited as opportunities for offloading computations and returning results back to the source node. These type of works typically target opportunistic connectivity scenarios Conti et al. (2010), i.e., where SMDs contacts occur from time to time. The focus of such RA mechanisms is on searching for time-variant routing paths between SMD with the aim of increasing task computing parallelism, and in this way reduce tasks completion time and/or balance the energy spent by collaborating SMDs.

In contrast, works perceiving user mobility as harmful for effective resource exploitation are Singh and Raza (2017); Habak et al. (2015); Lee et al. (2014); Vaithiya and Bhanu (2012); Jang and Lee (2011); Ghosh and Das (2010); Park et al. (2003). The fact that nodes move away from the communication range of the component that performs the resource allocation and/or collecting results are seen as an

Simulation scheme	Description	RA works
Synthetic	Movement guided by a mobility model	Shi et al. (2016); Li et al. (2015); Chunlin and Layuan (2014); Furthmüller and Waldhorst (2012); Shi et al. (2012); Vaithiya and Bhanu (2012)
	Movement represented through user presence/absence information	Singh and Raza (2017); Viswanathan et al. (2015); Habak et al. (2015); Jang and Lee (2011); Park et al. (2003)
In-vitro	Movement profiled from real SMDs: Encounter traces, Wireless Local Area Network (WLAN) traces	Lee et al. (2014); Mtibaa et al. (2013); Shi et al. (2012); Murray et al. (2010)

Table 2.5: User mobility evaluation methodologies

important source of tasks failure. To deal with this issue, RA mechanisms employ criteria to measure SMDs service availability, mainly based on historical frequency of connection and disconnection events, length of connection sessions and/or movement patterns inferred using GPS information.

Finally, there are works Shi et al. (2016); Li et al. (2015); Chunlin and Layuan (2014); Furthmüller and Waldhorst (2012); Murray et al. (2010) that do not propose concrete actions to exploit user movement but evaluate the performance of the proposed RA mechanisms under varied user mobility patterns. The schemes used by these works -and those aware of the singularity- to implement user mobility are outlined in Table 2.5.

User mobility is implemented through simulation by following a synthetic or an in-vitro scheme. One type of synthetic scheme uses movements derived from an artificial mobility model, where trajectory shape -circular, linear, random-, number of stops, time between stops and speed range of an SMD are common parameters used to characterize the model. SMDs are then configured with a mobility model, an initial location within a 2D plane and a communication radio. The plane can be associated to the coverage range of some reference node where the resource allocator logic resides Chunlin and Layuan (2014); Vaithiya and Bhanu (2012) or to an area where SMDs establish ad-hoc links with other SMDs in their vicinities Shi et al. (2016); Li et al. (2015); Furthmüller and Waldhorst (2012); Shi et al. (2012). The other type of synthetic scheme adopted by RA works for considering user mobility is by representing only connectivity information of SMDs, i.e., presence/absence of SMDs within regions that are supposed to be from where the RA logic operates. The connectivity information can be represented by probability scores associated to events of connection and disconnection of SMDs Singh and Raza (2017); Jang and Lee (2011); Park et al. (2003), or through in/out time intervals Viswanathan et al. (2015); Habak et al. (2015).

Alternatively, the in-vitro scheme aims at reproducing user mobility through profiles derived from real SMD traces. The works that adopt this scheme are Lee et al. (2014); Mtibaa et al. (2013); Shi et al.

(2012); Murray et al. (2010). The scheme is called "in-vitro" by analogy with the experimental methodology of biological sciences where a process is performed on some part of an organism but from outside the organism. In this case, mobility traces (parts) extracted from real SMD users (organism) and used to evaluate the performance of a resource allocation mechanism (process). The traces basically provide data of SMDs connectivity frequency and duration. When connectivity is with regard to other SMDs, for example due to people social interactions, the traces are called *encounter* traces. Popular encounter traces were those originated in the Haggle project, the MIT reality mining project and IEEE Infocom 2006 conference. These, and many others traces are available from the CRAWDAD website Kotz et al. (2009). There are also individual efforts Loke et al. (2015); Shi et al. (2012) that built custom encounter traces from the controlled study of selected groups of people. When data represents connections to access points, the traces are called WLAN traces and represent the SMDs user movements through different Access Point (AP)s of public or private buildings or campuses Balazinska and Castro (2003). An example of this type of traces is that of the Dartmouth university campus Kotz and Essien (2005) also available from the CRAWDAD website.

2.3.2 Lack of ownership

The lack of ownership singularity, understood as the sharable condition of SMDs resources by different task submitters including SMD owners, has received very little attention by researchers yet. Resources such as CPU cycles for CPU-intensive tasks or network bandwidth for data-intensive tasks should not be considered dedicated to external tasks but shared with owner processes and applications for two reasons. One is to minimize the impact of resources fluctuation caused by owner applications on external tasks. Since SMD owners have higher priority over resources usage, the load introduced by SMD owner processes and applications causes resources availability to fluctuate. By ignoring the real resources availability, the completion time/rate of external tasks can be underrated. The other reason is to minimize the external tasks invasiveness on SMD owner's experience. This means preventing external tasks from overloading an SMD and causing its battery to be depleted without care of the usage context, i.e., owner's current and future needs, the existence of power plugs in the proximity to charge the SMD, and/or owner's willingness to share resources on specific situations. Advocating to a context-aware RA might integrate ideas from anticipatory mobile computing Pejovic and Musolesi (2015) where SMDs sensing capabilities are used to infer current SMD owner context, predict owner's future context and/or events and propose a framework to advice the user on future decisions. Table 2.6 outlines the RA mechanisms aware of and evaluated for the singularity.

There is one work Ghosh and Das (2010) with focus only on external tasks performance which considers resource fluctuation due to user load. The work assumes that the rate at which resources are consumed/utilized by all tasks in every SMD is known to the RA mechanism. The execution of tasks in SMDs is modeled as a preemptive priority queue where external tasks compete for computing and bandwidth resources with other $P - 1$ classes of tasks that represent owner applications. From each class of task, the RA mechanism assumes to know its priority, arrival rate and load. This information is utilized to estimate the queue delay of external tasks on every node.

Approach	Focus	Objective	RAworks
Aware	Improve external tasks performance	Exploiting resources to reduce external tasks failure rates (or to increase tasks completion rates) in presence of resource fluctuation	Ghosh and Das (2010)
	Balance external tasks performance and SMD owner's experience	Exploiting resources while being aware of the future resource needs of the SMD owner	N/A
Evaluated		Add realism to the evaluation of the proposal	Rodriguez et al. (2014)

Table 2.6: Approaches for addressing lack of ownership

As Table 2.6 shows, there are not RA mechanisms which consider lack of ownership with focus on external tasks and SMD owner's experience. There is, however, one work Rodriguez et al. (2014) that considers the singularity within the variables of the experiments to evaluate the proposed RA mechanism. The evaluation methodology of these works and Ghosh and Das (2010) are outlined in Table 2.7.

Simulation scheme	Description	RA works
Synthetic	Non-available resources simulated via arrival rate of SMDs owner tasks	Ghosh and Das (2010)
In-vitro	Non-available resources simulated via CPU profiles	Rodriguez et al. (2014)

Table 2.7: Lack of ownership evaluation methodologies

As in the case of user mobility, the methodology utilized for including lack of ownership singularity within the experimental evaluation is simulation. A synthetic simulation scheme is adopted in Ghosh and Das (2010) where external tasks processing capability of SMDs is modeled as a portion of the maximum processing capability of SMDs that does not vary in time. Different SMDs are associated with different portions.

In (Rodriguez et al., 2014), it is employed an in-vitro scheme to simulate non-dedicated CPUs of SMDs. The CPU availability of each SMD is modeled through the configuration of a non-free CPU usage profile. A non-free CPU usage profile represents the CPU usage that an SMD used to execute owner processes and/or applications. When simulating dedicated scenarios, the CPU usage configured as base profile represents the CPU usage derived from the execution of the OS, which fluctuates around [2-10]% depending on the SMD model with the same OS version. In non-dedicated scenarios, the CPU usage fluctuates around 30% and represents the CPU cycles derived from the interaction of an average SMD owner who performs activities such as surfing the Web, checking emails, using a chat application, among other non-intensive CPU tasks. The fluctuation was not randomly generated but obtained by means of a

Approach	Focus		RA works
Aware	Conservative resource exhaustion	Pure	Furthmüller and Waldhorst (2012); Comito et al. (2011)
		Light	Viswanathan et al. (2015); Ghasemi-Falavarjani et al. (2015); Mtibaa et al. (2013); Shi et al. (2012)
	Greedy resource exhaustion		Singh and Raza (2017); Chunlin and Layuan (2014); Birje et al. (2014); Rodriguez et al. (2014); Wei et al. (2013); Rodriguez et al. (2010)
Evaluated			Shi et al. (2016); Loke et al. (2015)

Table 2.8: Approaches for addressing exhaustible resources

profiling process with an Android application that runs in the background on real SMDs.

2.3.3 Exhaustible resources

Table 2.8 outlines the RA mechanisms that address exhaustible resources. The works aware of the singularity are focused either on a conservative or a greedy resource exhaustion policy. Those in the first group aim at extending, as much as possible, the time of the first SMD leaving the network due to battery depletion. In those works, the concepts of fairness Jain et al. (1984) and/or network lifetime are commonly used to measure the quality of RA decisions. Works that promote a conservative resource exhaustion can be, in turn, pure conservative –when the extension of the network lifetime has top-priority over other criteria used in the RA decisions- or light conservative -when other than the aforementioned criterion, e.g., accomplishing tasks deadlines, are prioritized-.

An example of pure conservative RA mechanism is Comito et al. (2011) where a bottom-up strategy guided by nodes residual energy is used for assigning tasks. The scheduling logic is hierarchically organized in a set of interconnected cluster heads, each one in charge of the local scheduling decisions within their vicinity. The head of the nodes cluster from where a task is submitted tries to schedule the task within SMDs of the cluster. If no SMD of the local cluster is able to execute the task with its available energy, the task scheduling is delegated up to an inter-cluster scheduling level where the task execution is evaluated by the other cluster heads of the network. The less affected cluster –in terms of energy spent– by the execution of the task is assigned with the task. Other examples of pure conservative approaches are the service selection strategies called “Fair spending” and “Remaining charge” presented in (Furthmüller and Waldhorst, 2012). The Fair spending strategy distributes tasks (service requests) in such a way each SMD spends the same amount of energy, while the Remaining charge strategy selects the SMD whose remaining charge is less affected by the task execution.

By contrast, light conservative RA mechanisms do not have energy conservation as the top-priority objective but also pursue other QoS-related objectives, e.g., violating as few tasks deadlines as possible. Examples of these RA mechanisms are Ghasemi-Falavarjani et al. (2015); Viswanathan et al. (2015),

which propose multi-objective optimization approaches that aim at balancing tasks completion within tasks deadlines and maximization of the minimum residual energy of SMDs. In Mtibaa et al. (2013), tasks energy consumption is evaluated on every candidate SMD at one-hop and two-hops distance from the node willing to offload a task. The offloading cost is compared to the local execution and the option that maximizes the residual energy and allow the task to be completed within its deadline is selected. Similarly, the energy-aware offloading version of heuristics presented in (Shi et al., 2012) target the minimization of an utility function that contemplates tasks energy consumption on every candidate node and nodes residual energy without ignoring task deadlines. Notice that in these works the extension of SMDs lifetime is took into account upon every task assignment, provided this does not compromise the tasks execution deadlines.

On the other hand, greedy RA mechanisms adopt a resource exhaustion handling scheme that is not focused on making the whole distributed system last longer, but in exploiting resources to achieve the highest profit, the highest throughput, or the lowest tasks completion times. In those cases, RA mechanisms are aware of SMDs resource availability at the time of evaluating the feasibility of RA decisions. Examples following this scheme are Chunlin and Layuan (2014); Birje et al. (2014), where resources utilization is ruled by a market-based model where SMDs resources are offered at prices that resource consumers pay to the distributed system in exchange of certain QoS. By taking into account the resource providers energy constraints, the RA decisions must assure the QoS expected by the resource consumer while maximizing the profit. Other examples which follow a greedy resource exhaustion are Singh and Raza (2017); Rodriguez et al. (2014, 2010): they predict the resource quantification using information of remaining charge reported by the OS Application Programming Interface (API) to decide the number of tasks each resource provider should be assigned and maximize the amount of executed tasks that the distributed system completes. In (Wei et al., 2013), resource exhaustion is not inferred from SMDs remaining energy. The RA is modeled as a multidimensional 0-1 knapsack problem, where each SMD provides a fixed countable units of each resource type. For instance, resource provider p provides u units of resource type r. Hence, given the available amounts of each type of resource each SMD provides, the amount of resources each task requires, and the profit each task generates, the RA decisions are focused on maximizing the global profits of the distributed system, subject to the resources constraints of resource providers.

RA mechanisms prioritizing the utilization of energy-efficient resources, e.g., the “Minimum energy” strategy of Furthmüller and Waldhorst (2012), or ERRA Shah (2015), do not qualify neither as conservative nor greedy resource exhaustion approaches because RA decisions are not evaluated based on the current and/or future impact on residual energy. This causes that the most energy-efficient resource providers leave the system very early compared to the less-efficient ones.

Table 2.9 summarizes aspects concerning exhaustible resources. One of the aspects is the methodology used to reflect the resource exhaustion of SMDs itself, while the other aspect refers to the inclusion of baseline resource consumption that encompasses consumptions other than those directly caused by tasks execution, such as energy consumed by SMD owner interaction, background processes, OS runtime, connectivity maintenance, RA execution and administrative tasks (e.g., SMDs/tasks status messages updates), among others.

Aspect	Details			RA works
Resource exhaustion evaluation methodology	In-vivo laboratory tests			Ghasemi-Falavarjani et al. (2015); Viswanathan et al. (2015); Loke et al. (2015); Shi et al. (2012); Rodriguez et al. (2010)
	Simulated	Synthetic, guided by:	Tasks energy consumption	Wei et al. (2013); Chunlin and Layuan (2014)
			Resource energy consumption	Singh and Raza (2017); Vaithiya and Bhanu (2012); Ilavarasan and Manoharan (2010)
		In-vitro, guided by:	Tasks energy consumption	Viswanathan et al. (2015); Mtibaa et al. (2013); Shi et al. (2012); Comito et al. (2011)
			Resource energy consumption	Rodriguez et al. (2014); Furthmüller and Waldhorst (2012)
Consideration of baseline consumption	Yes			Ghasemi-Falavarjani et al. (2015); Viswanathan et al. (2015); Loke et al. (2015); Hirsch et al. (2016); Rodriguez et al. (2014); Furthmüller and Waldhorst (2012); Shi et al. (2012); Comito et al. (2011); Rodriguez et al. (2010)
	No			Chunlin and Layuan (2014); Mtibaa et al. (2013); Wei et al. (2013); Vaithiya and Bhanu (2012); Ilavarasan and Manoharan (2010)

Table 2.9: Implementing aspects of resource exhaustion singularity

Since SMDs resources provision is performed with finite energy sources, it is relevant to faithfully reflect the energy consumption of the resource utilization in order to avoid obtaining from an SMD more or less resources than it is actually able to provide. One methodology to represent resource exhaustion is through in-vivo laboratory tests, which suggests that energy consumption is directly experienced by the real SMDs that compose the experimental testbeds Ghasemi-Falavarjani et al. (2015); Viswanathan et al. (2015); Loke et al. (2015); Shi et al. (2012); Rodriguez et al. (2010). This methodology involves the setup of a wireless network for allowing SMDs to communicate, the installation of a middleware on SMDs from where the resource allocation logic operates, and the execution of a real/artificial parallel task-based application on SMDs. By nature, the methodology includes some baseline energy consumption, e.g., those derived from basic connectivity maintenance chores, SMDs OS and middleware runtime including RA administrative chores such as querying job status, remote nodes status monitoring, among others.

An alternative methodology commonly adopted for representing resource exhaustion is through simulation, which can be performed by employing a synthetic or an in-vitro scheme. In either schemes, the approaches for decreasing resources availability can be guided by tasks or resources energy consumption information. In the first case, tasks are associated to fixed amounts of energy consumption values, one for each type of resource utilized during task execution. For instance, the amount of energy consumed by I/O operations, CPU utilization, etc. Since SMDs present heterogeneous hardware performance, these amounts are different for each candidate SMD where a task could execute. For simulating resource exhaustion with this approach, SMDs energy -or battery capacity- is decremented according to the energy waste value associated to the task an SMD is assigned with. In synthetic schemes, as in (Chunlin and Layuan, 2014; Wei et al., 2013), the tasks energy waste values are randomly defined. In in-vitro schemes Viswanathan et al. (2015); Mtibaa et al. (2013); Shi et al. (2012); Comito et al. (2011), energy waste values are extracted by profiling tasks on real SMDs. For example, in Viswanathan et al. (2015), the energy consumption of a distributed object recognition application was profiled in seven SMDs including smartphones, tablets and laptops. In (Mtibaa et al., 2013) a power monitor hardware was used to profile the energy waste of different combinations of data transferring and computing operations in two smartphones. In (Comito et al., 2011), the authors derived the average energy waste of a data mining algorithm running over a dataset of 800 Kbytes in a smartphone. In (Shi et al., 2012), one set of experiments results are reported based on energy waste values derived from profiles of a face detection application and a speech-to-text application executed on two smartphones.

When simulation is guided by resource energy consumption, resources utilization units are associated with energy consumption units (e.g., Joules per MIPS, Joules per KB transferred) and tasks are associated to resource utilization units (e.g., amount of float-point operations, amount of input/output data). With this approach, tasks requirements are specified independent of SMDs hardware capabilities. For reflecting resource exhaustion, energy of SMDs is decremented conforms the value resulting from $TaskResourceUtilizationUnits * SMDEnergyWastedPerResourceUtilizationUnit$. In synthetic schemes guided by this approach, e.g., Singh and Raza (2017); Vaithiya and Bhanu (2012); Ilavarasan and Manoharan (2010), energy consumption is expressed as a fixed rate per resource utilization unit, and those rates are randomly defined. By contrast, in works using in-vitro schemes Furthmüller and Waldhorst (2012); Rodriguez et al. (2014), resources energy consumption is profiled.

In (Rodriguez et al., 2014), the profiling procedure derives in energy consumption traces and involves sampling energy level drops reported by the OS while maintaining an specific level of CPU utilization. The sampling starts with a fully-charged SMD battery and finishes when the SMD shuts down due to battery depletion. The procedure is repeated for several CPU utilization levels that represent idle, medium and full usage CPU states. The traces are used by a simulation software to imitate the energy consumption caused by the execution of CPU-intensive tasks, as well as the baseline energy consumption derived from SMD OS execution. In (Furthmüller and Waldhorst, 2012), the resource profile, called by the authors the *energy model* of an SMD, encompasses several resources. The profile is built from a set of benchmarking tests that exercise different resources usage combinations. Each combination is expressed as a linear equation that models the energy value obtained through the measurement of the energy consumed at benchmark execution time. By solving the linear equation system derived from all tests, the energy consumption rate of each resource under consideration is obtained. Baseline energy consumptions derived from OS processes are included in these rates. The simulation software uses SMD energy models to reflect tasks energy consumption that are described by a resource demand vector that indicates the quantities of each resource needed.

2.4 Discussion

In Section 2.1.1, the most frequently networking supports adopted by state-of-the-art RA proposals were presented. In this sense, it was observed a balanced amount of efforts targeting infrastructure-based and ad-hoc networks. Despite both network supports treat SMDs as resource providers and provide solutions in line with the singularities presented in Section 2.1.2, there are notable differences regarding how these have been addressed and evaluated.

In principle, RA efforts targeting infrastructure-based networks cover more singularities combinations than the efforts targeting ad-hoc networks, while these latter are exclusively focused on singularities combinations that include user mobility and resource exhaustion. Particularly, user mobility is a topic that has been extensively studied in communication research areas Elmangoush et al. (2015); Akyildiz et al. (2004); Camp et al. (2002), however not from the same perspective presented here.

When analyzing how user mobility is addressed, it is found that works targeting ad-hoc networks, particularly OppNets and DTNs, attribute the singularity a beneficial effect for resource allocation. Their strategy is centered on exploiting SMDs movements and future locations as opportunities for offloading computations to nearby SMDs and routing results back to the requesting node.

Works targeting infrastructure-based networks tend to associate user mobility with harmful effects. Their strategy for mitigating such effects is centered on quantifying connection reliability/availability supported on SMDs information of connection/disconnection events or SMDs movement parameters –such as direction and speed– that are individualized for each SMD. A missing feature in several works following this approach Lee et al. (2014); Vaithiya and Bhanu (2012); Park et al. (2003) is that the criteria employed to differentiate candidate SMDs do not include computing-related capabilities of SMDs, which limits the applicability of the criteria to homogeneous arrangements of SMDs. This is not the case

of Ghosh and Das (2010) where SMDs computing capabilities are considered in the resource allocation phase. The phase operates isolated from the technique used to select candidates SMDs which aims at mitigating user mobility effects. Other missing feature that several authors recognize as a strong limitation of their proposals is the exclusion of resource exhaustion-related indicators to complement the criteria used to rate the availability of SMDs under the control of a centralized resource allocation component. Some recent proposals Singh and Raza (2017); Viswanathan et al. (2015); Habak et al. (2015) addressing mobility-related issues also provide some treatment for the resource exhaustion singularity, which will be discussed hereafter.

As pointed out by several researchers in the area Fernando et al. (2013); Furthmüller and Waldhorst (2010), considering resource exhaustion is a top-priority goal when designing RA mechanisms that include battery-driven resource providers, like SMDs. In relation to how the singularity has been addressed, there is a tendency of RA proposals for infrastructure-based networks to adopt a greedy resource exhaustion policy, while those for ad-hoc networks tend to use a conservative resource exhaustion policy.

The usage of a conservative policy has a relation with an objective that traditionally characterized ad-hoc communication and sensor networks research areas, namely maintaining network reachability. Depleting the energy of a hub node in an ad-hoc network very quickly means not only loosing that node as resource provider but also potentially loose connection with a whole subgroup of resource providers. This problem is likely to affect purpose-specific scenarios where success strongly depends on the presence of all SMDs that integrate the distributed system, e.g., in rescue missions, surveillance operations, military units coordination, among others. A similar problem is observed in OppNets and DTNs where the adoption of conservative resource exhaustion policies is necessary to achieve the best usage of communication channels originated from intermittent SMDs encounters. In these types of networks, SMDs act as task executors and task carriers due to their movements. When two nodes make contact, i.e., two nodes are within their communication range for some time interval, an opportunity for allocating tasks and/or forwarding tasks results arises. Scheduling the tasks that a node should execute, or the tasks a node should store, carry and forward, demands the usage of conservative resource exhaustion policies.

By contrast, the usage of greedy resource exhaustion policies is often found in RA proposals targeting infrastructure-based networks. The application scenarios of these type of proposals are governed by different rules from ad-hoc networks application scenarios. In principle, SMDs are not expected to play the role of tasks and message forwarders to other SMDs, so less energy is spent on network maintenance duties. In other words, RA mechanisms for infrastructure-based networks operate with a wide view of available resources, which relates to the fact that communication with SMDs is usually assumed to be through single-hop links and with the support of dedicated communication infrastructure plugged to the electricity grid. Such one-to-one relationship between the RA component and the available resources turns the utilization of an SMD less dependent from the utilization of other SMDs, i.e., the resource exploitation is more loosely-coupled than in ad-hoc scenarios. Application scenarios of infrastructure-based networks are found in a wide range of domestic situations and public places where some sort of back-end communication infrastructure, e.g., cellular antenna or WLAN access point, is available.

In respect to the issues derived from the lack of ownership singularity of SMDs, there is a small subset of works Rodriguez et al. (2014); Ghosh and Das (2010) that superficially consider the singularity when

allocating resources. Any stronger effort in line with this singularity would help to better understand how to properly deal with the singularity, not only for improving the performance of external tasks execution but also to avoid deteriorating SMD owner experience. The proposal in (Ghosh and Das, 2010) focuses exclusively on external tasks performance and heavily relies on exact knowledge of SMDs internal task rates to operate. The authors recognized that such information is highly volatile and they plan to study impact of the refreshing period including network delay parameters. Another weakness of such proposal is its evaluation, since it does not use specific information of mobile user internal tasks rate, but fixed servers internal tasks rate. In (Rodriguez et al., 2014) no specific criteria is proposed to contemplate issues related to lack of ownership, but a simplified study of the effect of SMDs owner load on tasks completion rate is performed. The evaluation comprises simulations of non-dedicated SMDs with a constant CPU usage of 30%, which represents the typical load of an average mobile device user. All in all, there is plenty of room for improvements with regard to this singularity, not only in terms of indicators for RA criteria -with focus on either external tasks performance solely, or external tasks and owner tasks performance-, but also in the techniques used to capture and model SMD user-inflected load.

After analyzing the way works evaluate their proposals, it is found that few of them Loke et al. (2015); Ghasemi-Falavarjani et al. (2015); Shi et al. (2012) perform tests on real SMDs, while the majority use simulation software. Testing with real SMDs, i.e., in-vivo tests, has the benefit of observing how resource allocation decisions impact on the SMD hardware whose state is expected to change by such decisions. As a consequence, results are the closest to what one could observe in reality if the same experimental conditions were recreated. However, recreating the experimental conditions is the main drawback of in-vivo tests, due to it is extremely hard, when not impossible, to set the numerous variables that affect each element -nodes and links- of a distributed computing environment. For this reason, simulation is a widely accepted evaluation practice in the research of RA mechanisms comprising SMDs as resource providers.

Works targeting ad-hoc network scenarios use extensions of well-known simulation software, such as the OMNet++[1] or the NS-3 simulator [2], which were developed for simulating data routing algorithms, nodes movement and communication ranges in mobile networks. Several works targeting infrastructure-based scenarios that evaluate their proposals through simulators do not make the software available, and thus reproducibility of experiments is compromised.

Other works, by contrast, employ software traditionally used in Grid Computing research, e.g., GridSim Buyya and Murshed (2002), SimGrid Casanova (2001), extensions of JavaSIM[3], among others. A further discussion on simulation models for mobile Grids is presented in Section 4.2 of Chapter 4. In short, the strengths of the aforementioned softwares reside on the capabilities for simulating different nodes processing rates and, in some cases, nodes mobility patterns, links capacities, different types of task relationships, i.e., dependent and independent. Besides, some of these simulators have been extended to simulate exhaustible resources, but in every case, it is done through oversimplified synthetic models with no support for dealing with real energy consumption traces. Another missing feature of

[1] https://omnetpp.org/
[2] https://www.nsnam.org/
[3] http://javasim.ncl.ac.uk>

current simulators is the lack of models to represent SMDs owner activities.

Last but not least, from the above analysis it is clear that, irrespective of the networking support adopted, singularity/ies combination targeted and the methodology used for evaluation, most state-of-the-art RA mechanisms need to be fed with highly accurate knowledge of tasks requirements to operate. In other words, the performance of RA mechanisms heavily depends on accurate descriptions of tasks requirements, i.e., resources needed. Task requirements are assumed to be provided typically by the user who creates the task and the forms of indicating them is through processing time, amount of CPU cycles, million of instructions or an equivalent value expressed in Joules spent on execution. This limits the applicability of many RA mechanisms. In part, this is because, in practice, tasks historic execution time is not always available to be used as scheduling input. Moreover, in the general case, to predict task execution time it is firstly necessary to know whether the task actually ends, which means solving the halting problem Wilhelm et al. (2008). Assuming that tasks execution time is known by the RA mechanism, the requirement of knowing the energy that tasks consume is another barrier that limits the applicability of many proposals. Precise models to estimate battery consumption are based on complex differential equations Chang (2013); Hu and Yurkovich (2012), and solving them represent a computationally complex task itself.

Chapter 3

A two-phase scheduling approach for CPU-bound jobs

3.1 Motivation

Wireless infrastructures to which people connect with SMDs are present in an increasing number of daily life in public and private places. Coffee shops, restaurants, shoppings, university campuses, work offices, just to mention few examples, are equipped with WLAN access points to let people surf the Web, check emails, read newspapers, play online games, or simply to stay connected for receiving messages and notifications of their interest. These contexts represent potential resource exploitation opportunities where aggregated computing capabilities of SMDs are available to solve complex computational problems.

Taking advantage of the wide view and centralized control of resources offered by infrastructure-based networks, and foreseen proper incentives that encourage people to share their unused computing capabilities of SMDs Restuccia et al. (2016); Duan et al. (2014), the outcome of such system heavily depends on the criteria that the RA mechanism utilizes to distribute jobs.

From the extensive literature review presented in the previous Chapter it was noticed that current RA efforts have several limitations, namely:

- Current RA proposals targeting distributed computing using SMDs suffer from requiring hard-to-obtain information of tasks in order to operate.
- There are no works proposing *easy-to-implement* guidelines for greedily exploiting SMDs computing capabilities while coordinating nodes via infrastructure-based networks.
- Simulation is the most accepted evaluation practice in the research field, however existing simulation tools do not allow to faithfully reflect battery consumption of SMDs derived from the execution of CPU-intensive tasks in a more realistic way other than synthetic models.

- None of the existing simulation tools used for evaluating RA mechanisms of CPU-intensive tasks in mobile Grids model the non-dedicated nature of SMDs using data of CPU utilization derived from real SMDs owner interaction. By considering that owner interaction do not only influences tasks execution time but also the rate at which energy from battery is consumed, including this feature is necessary to advance the research and evaluation of new RA proposals that consider SMDs as CPU *cycles providers*.

3.2 Problem statement and hypothesis

The above limitations represent the motivation for the present thesis. In this way, the scheduling problem and conditions that guides the RA approach proposed in this thesis is described as follows: *Given a set of CPU intensive tasks of varying size and a set of heterogeneous SMDs in terms of battery capacity and computing capabilities, find a mapping of tasks to nodes to allow the distributed computing system to finalize as much tasks as possible with the available energy*. The resource allocation is subject to the following conditions:

1. Neither individual nor the whole set of tasks execution time, number of Mega Float-point Operations (MFLOP)s required for completion or energy units consumed are known in advance to the scheduling logic.

2. Tasks have no parallel dependencies nor priorities, and are scheduled by following an online scheme.

3. The scheduling component has a buffer of size zero, meaning that a decision for every task needs to be taken as soon as each they arrive to the scheduling component, sending the corresponding task input to the selected node. Such condition comes from the fact that SMDs resources are available for a limited time.

4. Scheduling decisions taken for tasks arrived at time t are not affected by the arrival of a new task, nor the completion of already mapped tasks at time $t + \alpha$. It means that tasks re-scheduling is not allowed unless an idle SMD is detected.The work hypotheses on which this thesis is based on were mentioned in Chapter 1 and are detailed again below:

Hypothesis 1: Task scheduling criteria that estimate future SMDs potential computing capabilities using energy-related factors achieve better throughput than criteria that do not use it.

Hypothesis 2: The unknown tasks requirements, inaccurate estimations of SMD computing potential and unknown tasks requirement partially break previous energy-aware task scheduling decisions creating sub-exploited slots of computing cycles that can be exploited through a dynamic task re-balancing mechanism.

Hypothesis 2.1: The improvement achieved by such re-balancing mechanism is conditioned by the energy-aware scheduling decisions made in the past, namely the first scheduling phase.

3.3 The proposed approach

Motivated by the challenges posed by the idea of considering SMDs as first class citizens of computing power and the limitations of current research efforts in the area, in this thesis I propose a two-phase approach for scheduling CPU-intensive tasks in infrastructure-based mobile Grids subject to the conditions 1-4 described in Section 3.2. The approach addresses issues from a singularity combination that has been little explored in state-of-the-art works, which is the exhaustible resources singularity combined with lack of ownership singularity of SMDs.

The awareness of the approach with regard to exhaustible resources is through the criteria that the scheduler uses to estimate SMDs computing capability. Besides, since the objective of the scheduling approach is to achieve the highest amount of finalized tasks per energy unit, it falls into the category of those works focused on greedily exploiting exhaustible resources.

Moreover, the proposed approach addresses the lack of ownership singularity through an in-vitro scheme implemented using information of real SMD users interaction sessions. Exhaustible resources is also implemented through an in-vitro scheme guided by resource energy consumption, i.e., with energy consumption profiles extracted from real SMDs. Details of the singularities implementation materialized in a simulation tool, and the experiments that validate its functioning are provided in Chapter 4.

The strengths of the approach are three-fold: the singularities combination it addresses, the dual in--vitro evaluation methodology it employs, and its practical applicability. The last strength is due to the scheduling approach does not depend on hard to obtain tasks information to operate, but on easy to obtain SMDs information.

As stated at the beginning of this Section, the proposed scheduling approach is structured in two phases. The first phase activates as soon as a task arrives to a central component called *proxy* that maintains an active one-hop wireless connection to every candidate SMD that offer its computing resources for task execution. The candidates are ranked based on the criterion of the scheduler and the best ranked SMD is assigned with the task.

Figure 3.1 depicts a proxy-based mobile Grid environment composed by Fixed Virtual Resource (FVR) and Mobile Virtual Resource (MVR) integrated by fixed computers and SMDs, respectively. In this context, this thesis focuses on the study of intra-MVR battery-aware local schedulers. A Virtual Resource is seen as a single node by the entire Grid environment and differs from a traditional cluster mainly because it could be composed by *heterogeneous* hardware. Offering local resources behind a proxy to a Grid environment is a strategy commonly adopted by traditional Grid platforms such as Ibis-Satin van Nieuwpoort et al. (2010) and GridGain (`www.gridgain.com`).

With every new task arrival, the list of candidates is ordered using novel battery-aware criteria. Different criteria are proposed to rank SMDs computing capability which are presented in Section 3.3.1.1, Section 3.3.1.2 and Section 3.3.1.3. The criteria, which are easy to implement with current technologies Hirsch et al. (2017c), combine static information of SMD computational characteristics, dynamic information of available SMDs energy and currently queued tasks.

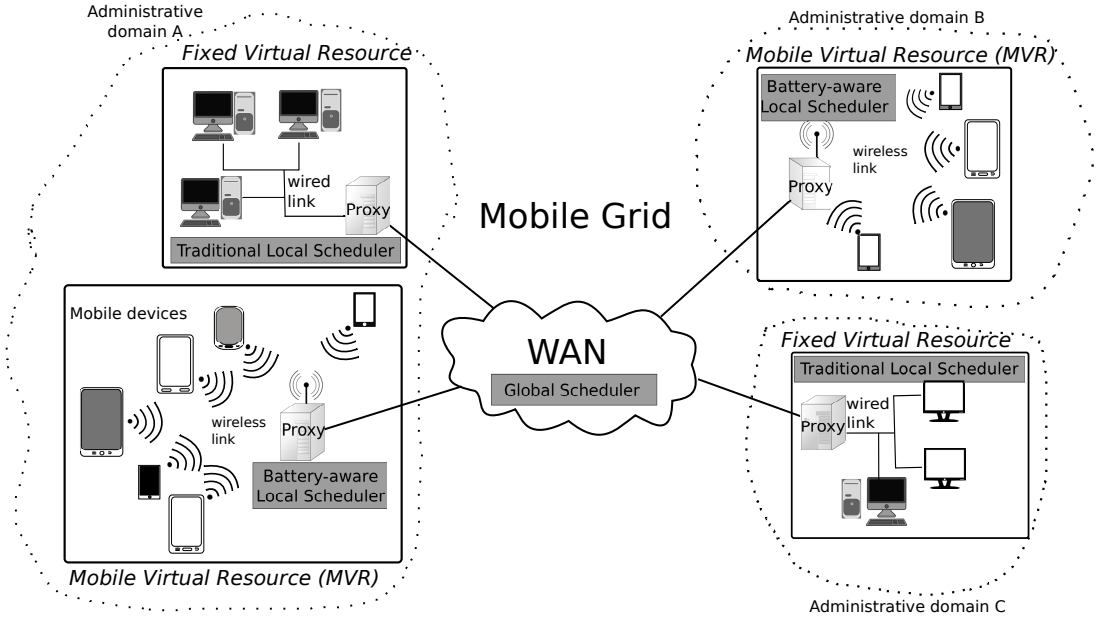


Figure 3.1: Proxy-based Grid environment

Tasks information is assumed to be unknown by the battery-aware criteria. Moreover, tasks execution time is assumed to vary from node to node. Intuitively, SMDs assigned with a subset of tasks will finish their assignments, while others will drain their batteries leading to queued tasks whose execution will not ever start. Such effect may also be caused as consequence of SMD owner interaction. Irrespective of the case, the energy of idle SMDs, namely the computing capability, would be wasted if tasks load is not re-balanced. The second phase of the proposed scheduling approach aims at dealing with such potential waste of computing capabilities. It employs job stealing techniques, which has been successfully applied in traditional Grid and cluster computing middlewares[1] van Nieuwpoort et al. (2010). The second phase acts as a load re-balancing mechanism where SMDs play an active role in scheduling decisions. Figure 3.2 illustrates the two-phase scheduling approach. As soon as tasks arrive to the proxy, the first phase of the scheduling approach activates and each task is assigned to an SMD. As the time passes, and the tasks queue of an SMD empties, the second phase of the scheduling activates to trigger a load re-balancing session.

Section 3.3.1 deeps into details of how the first phase of the scheduling approach is materialized in a proxy-based network. Section 3.3.2 provides details of the second phase of the scheduling approach, i.e., the decentralized participation of SMDs in scheduling decisions through job stealing techniques.

3.3.1 Centralized first phase

The first phase of my scheduling approach consists in assigning every task submitted to the proxy to an SMD, which is selected based on a ranking performed over the candidate SMDs using energy-aware criteria. The adoption of a proxy-based networking support favors a registration process where SMDs

[1] `https://www.gridgain.com/api/javadoc/org/gridgain/grid/spi/collision/jobstealing/GridJobStealingCollisionSpiMBean.html`

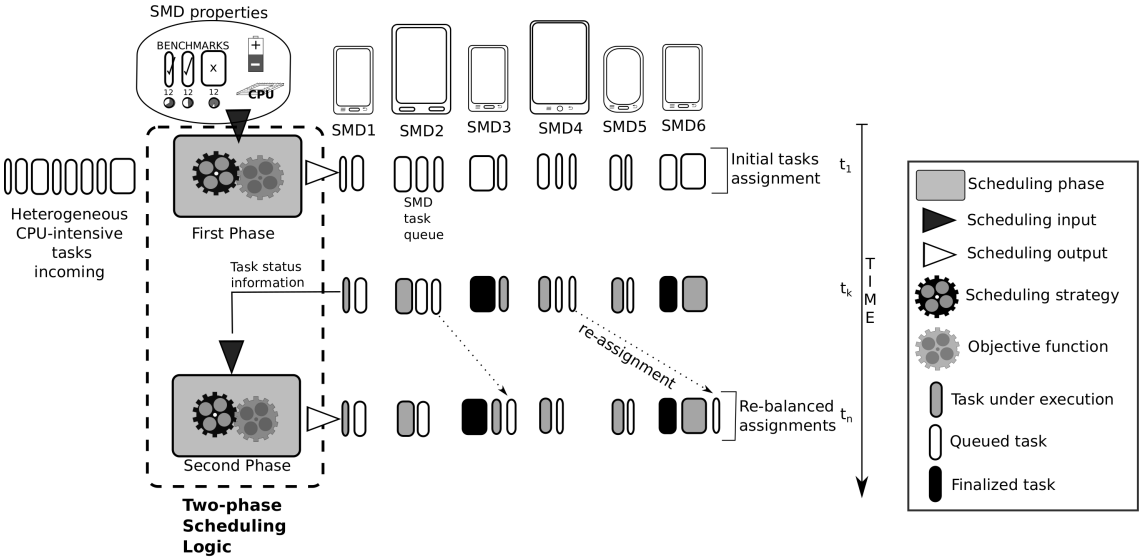


Figure 3.2: The two-phase scheduling approach for CPU-intensive tasks: A schematic view

willing to offer computing resources join the proxy Khalaj et al. (2010). That registration process is an opportunity for the proxy to identify and categorize all SMDs, e.g., in terms of computing capability. For this purpose, the approach associates each SMD a *feature profile*. The feature profile contains properties of an SMD, including benchmarks results and manufacturer data. The former group is composed by the scores derived from the execution of a benchmarking software –e.g. Linpack for Android[2] or SciMark 2.0[3]–, while the latter comprises information about the typical battery specifications declared by the device manufacturer. Same model devices are supposed to have similar feature profiles. Therefore, instead of generating a feature profile for each device upon proxy registration, all the devices of the same models might be represented under the same feature profile. The feature profile data is later employed by the proposed energy-aware criteria to rank SMDs computing capability.

Table 3.1 lists the investigated feature profiles (as columns) for different SMD models. The first row of each feature profile shows the computing capability of the device expressed in Mega Float-point Operations per Second (MFLOPS). These values are the average of twenty runs obtained with Linpack for Android. Another way of measuring the computing capability is by aggregating the quantity of a set of different well-known benchmarks –second row– that the SMD was able to perform within a certain time period. That value is presented in the third row, while the time period is the time the battery last from full charge (100%) to cut-off voltage (0%). The feature profile also includes energy-related properties, i.e., battery capacity as informed by the manufacturer. The last row presents a novel way of rating the computing capability of a device, which comprise benchmarks information with battery capacity information into a score called Job Energy Consumption Rate. This score is a constant used by the node ranking formula of the battery-aware criterion developed in Section 3.3.1.2.

Scheduling decisions based only on static properties of SMDs, i.e., read from the feature profile, would not contemplate resource exhaustion and actual load of an SMD. Then, dynamic information, i.e.,

[2] https://play.google.com/store/apps/details?id=com.sqi.linpackbenchmark&hl=en
[3] http://math.nist.gov/scimark2/about.html

		Samsung I5500	ViewSonic ViewPad 10s	Acer Iconia Tab A100
Mega FLOPS		7.60	17.07	61.66
Benchmark	Gaussian random generator	10	42	30
	Prime checker	10	42	30
	Fast Fourier Transform	11	43	30
	Sieve of Eratosthenes	10	43	30
	Towers of Hanoi	10	42	30
# Total Executed Benchmarks		51	212	150
Battery Capacity (µAh)		1,200	3,300	1,530
Job Energy Consumption Rate		23.53	15.57	10.20

Table 3.1: Feature profiles of SMDs

generated at runtime is also considered. Such dynamic information comprises the current State Of Charge (SOC) of an SMD, and historic workload stats, i.e., current number of assigned tasks to an SMD and average tasks execution time of an SMD.

Below, three alternatives for energy-aware criteria combining feature profiles and dynamic data developed in the course of this thesis are described. These criteria are the Enhanced Simple Energy-Aware Scheduler (E-SEAS) (Section 3.3.1.1), the Job Energy-aware Criterion (JEC) (Section 3.3.1.2) and the Future Work aware Criterion (FWC) (Section 3.3.1.3).

3.3.1.1 The Enhanced Simple Energy-Aware Scheduler (E-SEAS)

The E-SEAS is an improved version of the SEAS Rodriguez et al. (2010). It is built upon the combination of almost the same three components of the SEAS formula. The difference is the component through which the remaining uptime of the SMD is estimated. The formula applied for such estimation works well with notebooks and netbooks Rodriguez et al. (2012) -the kind of devices for which SEAS was initially proposed-. However, when trying to apply the same formula to estimate the remaining uptime of SMDs such as smartphones and tablets, it is required a considerable time (several minutes or even hours) until the uptime estimations become precise. This causes spurious ranks of SMDs which derives in chaotic scheduling decisions. Therefore, this thesis introduces a change in regards to how the SEAS handles the remaining uptime. The change derives in the E-SEAS that use the SOC in exchange of the $estimatedRemainingUptime$ component of SEAS. The E-SEAS SMD rank formula is:

$$SMDRank = \frac{SOC * flops}{assignedJobs + 1} \tag{3.1}$$

where SOC is the percentage of remaining battery charge which is accessible through the SMD OS API, $flops$ is equivalent to the $benchmark_m$ component of the SEAS formula, and $assignedJobs$ refers to the $number\ jobs_m$ component.

3.3.1.2 The Job Energy aware Criterion (JEC)

An SMD rank based on this criterion takes into account the relationship between the energy used and total of benchmarks an SMD is able to execute. These values are extracted from the device feature profile. Energy used refers to the battery capacity in milliamperes-hour as reported by the device manufacturer (Battery Capacity row in Table 3.1) while total of benchmarks refers to the quantity of established benchmarking tests that the SMD is able to complete before the battery depletion event Rodriguez et al. (2012) (# Total Executed Benchmarks row in Table 3.1). The quotient of energy used over total of benchmarks gives the rate of energy consumption per benchmark, or from now on, $jobEnergyConsumptionRate$ (Eq. *3.2*):

$$jobEnergyConsumptionRate = \frac{BatteryCapacity}{\#TotalBenchmarks} \tag{3.2}$$

The ranking formula defined by this criterion is presented in Eq. 3.3:

$$SMDRank = \frac{SOC}{jobEnergyConsumptionRate} * \frac{1}{assignedJobs + 1} \tag{3.3}$$

JEC combines runtime information (SOC and $assignedJobs$), and static properties associated to SMDs represented by the $jobEnergyConsumptionRate$ score. For example, in case two devices A and B have equal SOC and $assignedJobs$, but device B has lesser $jobEnergyConsumptionRate$ than device A, then the criterion gives a higher rank to device B than that of device A with the assumption that device B might better exploit its energy to execute tasks.

3.3.1.3 The Future Work aware Criterion (FWC)

This criterion, unlike JEC, is purely based on runtime information. It considers that the future available computational resources of an SMD could be estimated by analyzing the available computational resources the SMD had in the past. In other words, FWC assumes that the computing performance delivered by an SMD in the past is sustained in the future. Like for E-SEAS and JEC criteria, SOC is obtained through the battery events periodically reported by a SMD OS, and $assignedJobs$ is the number of jobs in the SMD queue. Moreover, the $averageTimeToCompleteJob$ value is calculated by the scheduler and represents the average time an SMD takes to complete each of its scheduled tasks. All in all, the ranking formula is defined in (Eq. 3.4):

$$SMDRank = \frac{SOC}{averageTimeToCompleteJob} * \frac{1}{assignedJobs + 1} \tag{3.4}$$

The left quotient of the formula penalizes high average task execution times while considers remaining battery percentage through SOC values, while the right quotient avoids leaving some SMDs less loaded than others. When comparing two SMDs with equal SOC and assigned jobs, the one with less

$averageTimeToCompleteJob$ value would be assigned a higher $SMDRank$ value. Consequently, the SMD with the highest rank value has the highest chance of being assigned a task.

A distinctive characteristic of this criterion is the cold start effect. While schedulers based on JEC and E-SEAS operate effectively from the first scheduled task, FWC based schedulers need some time to achieve effectiveness because FWC uses information of historical jobs finalization time to calculate the $averageTimeToCompleteJob$ value. At the time of determining the $SMDRank$ of a device, if no tasks have been finalized yet, but there is a task currently executing in that device, then the $averageTimeToCompleteJob$ is computed as twice the time the task has spent executing so far. If no tasks have been finalized yet, nor a task is executing in the device, a value of "one" is returned as $averageTimeToCompleteJob$ to prevent the left quotient from being invalid.

3.3.2 Decentralized second phase

The above energy-aware criteria uses runtime information to consider the potential availability of computing resources. However, the unknown computational requirements of tasks, and the lack of updated/accurate information, e.g., of future CPU usage due to owner activities affect the scheduling decisions of the first phase, making them perfectible as the time passes. A second phase –re-balancing phase– is important to adapt scheduling decisions made in the past to periodically changing resource availability in a MVR.

Job Stealing techniques have been effectively applied as a re-balancing mechanism in traditional distributed computing van Nieuwpoort et al. (2010) and also in mobile Grids Loke et al. (2015); Rodriguez et al. (2014). Basically, Job Stealing aims at minimizing unused SMDs in distributed environments, preventing tasks in an SMD from waiting to be executed when there are idle SMDs elsewhere. For the purposes of this thesis, this behavior in conjunction with a detailed study of the dynamics of an MVR inspired the formulation of the second scheduling phase, instantiated with Job Stealing.

In the approach, the use of Job Stealing involves configuring a victim selection strategy and a stealing policy, i.e., the SMD from which tasks will be stolen and the quantity of tasks to steal respectively. Steals are performed from an idle SMD or *stealer*. One victim selection strategy and one stealing policy represent a particular configuration of the Job Stealing technique. Given the number of possible configurations, this set is shortened by selecting the best configurations in terms of throughput explored in similar earlier works (Rodriguez et al., 2014). Then, the considered victim selection strategies in this thesis are:

Best Ranking Aware Stealing (BRAS): This strategy selects the SMD with the highest ranking according to the criterion used. This strategy aims at offloading the least overloaded SMDs. As a result, victims are more likely to become idle and, in turn, they would be themselves able to take jobs from other SMDs. Basically, this strategy is expected to generate an offloading chain reaction.

Worst Ranking Aware Stealing (WRAS): Instead of selecting the best ranked SMD, it selects the worst ranked one. In this case, the goal is to globally balance the load because idle SMDs take tasks from the most loaded SMDs.

Moreover, the next two stealing policies describe how a stealer determines the number of tasks it will try to steal upon each attempt. In practice, stealing several tasks at once might reduce the networking overhead because it requires establishing only one connection. Networking requires energy Ding et al. (2013) and reducing the need for that might extend battery life. For this issue, the policies considered in this work are:

Fixed Number: A stealer always steals the same (statically configured) number of tasks. For n-core SMDs, each steal attempt might actually try to steal n tasks (the number of tasks the SMD is able to physically execute in parallel according to the number of cores).

Exponential: The number of stolen tasks exponentially increases based on the number of times the SMD became idle. If an SMD becomes idle for the n^{th} time, it would steal 2^n tasks. For example, the first time, the stealer steals one task (2^0), the second time, the stealer steals two tasks (2^1), and so on.

An aspect that differentiates this Job Stealing instantiation with respect to that of Rodriguez et al. (2014) is the consideration of the networking energy cost inherent to a decentralized scheduling logic. By introducing Job Stealing as part of the scheduling logic, SMDs are more active in terms of data transferring because when an SMD finalizes all its queued tasks, Job Stealing looks for other SMDs to steal their queued tasks. By considering that, such decentralized scheme implies a communication overhead, due to SMDs sending stealing messages and eventually moving tasks from a victim's queue to the stealer's queue. The potential gains offered by this technique are evaluated by considering the networking energy overhead associated.

Chapter 4

Evaluation methodology

4.1 Preliminaries

As shown in the analysis presented in Section 2.3 of Chapter 2, in-vivo experimentation has been used to evaluate those RA mechanisms which address the resource exhaustion singularity only. In-vivo experimentation results in a slow, tedious, expensive and hard-to-reproduce evaluation methodology. Arranging tens or hundreds of real SMDs implies either encouraging real users to participate in experiments or acquiring a varying set of SMDs.

Moreover, to provide solid statistically-significant results, RA mechanisms must be tested many times against different scenarios varying amount of jobs, job sizes, job start times, small/medium/large mobile Grids, etc. With in-vivo experimentation, testing many times is slow, mainly due to the time it takes to charge and discharge batteries to prepare devices for the next experimentation round. Such time is to some extent known and controllable when using dedicated SMDs, but that is not the case for SMDs that belong to real users. In-vivo experimentation also suffers from the low reproducibility issue. Configuring or resetting experimental conditions is extremely hard to achieve because hardware state is under the control of the SMD owner. In addition, the well-known aging problem Takeno et al. (2005) of Li-ion batteries contributes to such difficulty. Charging and discharging SMDs repeatedly as a consequence of periodic execution of experiments might compromise the consistency of results over time, and of course, the lifetime of the batteries.

RA research in traditional distributed computing environments, i.e., Grid Computing or Cloud Computing, embraced simulation as evaluation methodology to reduce experimentation time and produce repeatable tests. GridSim Buyya and Murshed (2002) and the CloudSim Calheiros et al. (2011) toolkits evidence of this fact. To quantify this claim, according to Google Scholar, just the reference papers of both toolkits have received in conjunction over 5,000 citations. Like in traditional distributed computing, research in mobile Grids traverses similar challenges with regard to complex environments given by nodes heterogeneity, complex nodes configuration and behavior. For this reason, simulation is a commonplace evaluation methodology in the area.

Despite simulation is far the most practiced evaluation methodology in the area, there is much work to do with regard to models for simulating the dynamics of mobile Grids, specially concerning energy consumption caused by computing resources. Many RA efforts use software developed with a focus on modeling wireless connectivity issues (e.g., due to nodes mobility), or energy consumption caused by computing resources but by employing synthetic models that do not faithfully reflect resource exhaustion of real devices. Besides, none of the aforementioned software considers the lack of ownership singularity.

In this chapter, it is presented a simulation tool specially developed in the context of this thesis for investigating the performance of scheduling algorithms in proxy-based mobile Grid settings. The simulation tool goals are summarized as follows:

- Providing an end-to-end support for research in job scheduling algorithms that is focused on mimic the computing potential and resource exhaustion of SMDs caused by CPU usage derived from job execution and/or user interaction.
- Supporting variability of the infrastructure-based mobile Grid model adopted in the tool. Particularly, variability is considered for the following aspects: devices energy consumption, devices non-dedicated computing resources, and job characteristics. Devices energy consumption sub-model uses traces of real Android devices to mimic the energy consumption caused by CPU usage. Devices non-dedicated resources sub-model uses a third-party empirical probabilistic model to mimics the CPU usage caused by owner interaction. Lastly, the job characteristics, i.e., arrival time, CPU operations, input and output data size, are generated using statistical distributions.
- Complementing the computing/energy consumption sub-models with a networking model for reflecting resource exhaustion caused by data transferring operations between the proxy and devices.
- Offering an open source implementation, which is programmed in Java and available in GitHub[1]. The software is composed by modular software artifacts that include:
  - An object-oriented design to represent the dynamics of infrastructure-based mobile Grids which can be externally configured through configuration files.
  - A simple ad-hoc simulation engine that follows an integrated approach, i.e., combining process driven and event driven approaches Kesaraju and Ciarallo (2012) for running models.
  - An experimental methodology for researchers users, materialized in an application, for capturing energy depletion traces at discrete CPU usages from real Android devices.
  - In line with the above item, an application for creating traces to represent device user interactions by composing discrete CPU usage traces and empirical user interaction models.
  - Storage of simulation output in plain text logs, and in a lightweight relational database model for post-simulation analysis.

In Section 4.2 of this chapter, it is discussed related efforts concerning simulation in distributed computing. It is also argued why the software currently employed for simulating the dynamics of mobile

[1] `https://github.com/cmateos/mobileGridSimulator`

Grids do not fulfill the features required for evaluating the proposed schedulers. Section 4.3 discusses the functional features, the core design of the simulation model, and relevant implementation details of the software materializing it. Section 4.4 deeps into simulation input concerns. It provides an introduction of how to configure each component of a mobile Grid or MVR instance. Then, Section 4.4.1 details the Android application developed to capture battery consumption profiles from real devices, and Section 4.4.2 explains how battery consumptions profiles targeting discrete CPU usages are combined to simulate user activity. Profiles serve as input to the model when simulating specific mobile Grid instances. Section 4.5 shows the output capabilities of the simulation tool. Lastly, modeling correctness is assessed in Section 4.6.

4.2 Related efforts

From the distributed computing standpoint, SMDs were initially embraced by the MCC paradigm Fernando et al. (2013); Khan et al. (2015), a subset of Cloud Computing. MCC augments devices capabilities by using resources (mainly power and storage) in remote servers to overcome smartphone constraints and enhancing user's experience Khan et al. (2015). A related recent paradigm known as Fog Computing Gupta et al. (2017) follows the same idea as MCC but uses nearby servers, i.e. in the same local network. Moreover, Mobile Crowd Computing (MCrC) is a subset of MCC in which nearby devices can act as the destination "server" when offloading jobs or data from a surrounding device Fernando et al. (2016). Mobile devices willing to provide computational resources to other mobile devices are arranged into the so-called mobile Grids or mobile-edge Clouds, using an *infrastructure-based* or an *ad-hoc* networking support. The simulation tool presented in this chapter sits in the area of MCrC and particularly infrastructure-based mobile Grids. Far from being only different forms of arranging devices, there are substantial differences in how schedulers operate in infrastructure-based and ad-hoc networks that turn some assumptions in one research area not valid in the other and vice versa. The differences relate to the way information traverses the network.

In ad-hoc mobile Grids, for example, wireless data transferring occurs through multi-hop paths where mobile devices can play the roles of senders (clients), receivers (servers) and forwarders (routers) of data. In infrastructure-based mobile Grids, wireless data transferring occurs through one-hop links since mobile devices are assumed to be connected to a stationary access point or antenna playing the role of server. Then, in ad-hoc mobile Grids, due to the potential absence of a direct communication channel between client and servers, job scheduling effectiveness relies on how load among nodes is distributed for resources discovery and coordination, jobs data and results dissemination, which in turn heavily depends on the effectiveness of data routing algorithms. In infrastructure-based mobile Grids, the impact on energy of all these issues is minimized due to the scheduling component has the global view of all resources and direct communication with all mobile devices. Then, it is common in ad-hoc mobile Grids research to adopt existing network simulation frameworks, such as NS-3 and OMNET Chengetanai and O'Reilly (2015), which provide abstractions for modeling physical processes affecting wireless communications, namely node mobility, signal propagation Shah (2015) and collision, or even aspects associated to networking processes such as packet fragmentation, packet buffering, packet retransmissions.

All these features revolve around data transferring-related concerns, and leave outside computing-related concerns –e.g., how mobile device CPU usage affects energy consumption– that are relevant for research in schedulers for infrastructure-based mobile Grid.

Simulation frameworks with built-in support for modeling computing-related concerns are those conceived for Grid Computing research. Notably, GridSim Buyya and Murshed (2002) is an event-driven simulation toolkit that provides abstractions for modeling large-scale distributed computing systems integrated by million of resources with single or multi-processors, with shared or distributed memory managed by time or space shared schedulers. Among the recently added features are functionalities to read real workload traces taken from supercomputers. A more versatile option is SimGrid Casanova (2001), upon which specific models have been created for simulating a variety of distributed systems including clusters, content sharing in wide and local area networks, data centers, and Cloud environments. However, these models allow to simulate the behavior of dedicated computing resources connected through wired networks. Modeling non-dedicated computing resources is a crucial aspect for MCrC any simulator since mobile devices computing resources are shared with the owner.

SimGrid has also been used as base for volunteer computing simulation models, e.g., ComBos Alonso-Monsalve et al. (2017) which provides functionality for modeling the whole BOINC infrastructure including scientific projects, server components providing input files associated with project's jobs, clients donating CPU cycles of desktop computers and the scheduling component at the client side that schedules resources among different projects. The volunteer computing paradigm also deals with non-dedicated computing resources but ignores energy consumption concerns as resources are fixed (desktop computers). Modeling energy-aware resource schedulers is supported in simulators for Green Computing such as GreenCloud Kliazovich et al. (2012), but such simulators are not applicable to infrastructure-based mobile Grid research for several reasons. First of all, energy saving strategies in Green Computing involve operations which require special privileges over hardware resources, including dynamic shutting down of components, which is clearly unfeasible from the scheduling logic in mobile Grids due to the lack of resources ownership. Secondly, energy awareness does not imply being aware of finite energy but using less energy or dissipating less heat for computing resources which are plugged to the electricity Grid, while in mobile Grids computing resources rely on energy supplied by batteries.

All in all, extending any of the above frameworks for providing the missing functionality necessary for modeling the computing resource provision of infrastructure-based mobile Grids might lead to much more effort than building a new simulation tool designed to support the specific features, which represents a side contribution of this thesis. Besides, from the usability perspective, by extending some of the above frameworks, the modeler might be in the necessity of configuring components that are not relevant to the problem being modeled. To make things worst, by running models with many irrelevant components, valuable modeling time and memory/CPU cycles for running heavy model implementations could be wasted. This motivates the development of this simulation tool, which provide a model for simulating jobs with CPU and data transferring requirements using mobile devices as resource providers and coordinated through a central component called proxy where scheduling takes place.

4.3 Modeling and simulating infrastructure-based mobile Grids

This section deeps into the core of the model built and simulation related aspects. The content is divided in two sections. Section 4.3.1 presents the conceptualization and scope of the model, where the reader will find the level of detail with which domain specific aspects are modeled. Section 4.3.2 provides details of structural and dynamic views of the simulation tool, including components of the execution engine for loading and running models, and the infrastructure-based domain specific components developed from scratch.

4.3.1 Defining the conceptual model

As suggested in simulation practice literature Robinson (2014); Sokolowski and Banks (2010); Banks et al. (2005), I start by defining the conceptual model of the system. Firstly, it is described the objective of the model, its inputs (also known as experimental factors), outputs (reports), assumptions and then, the main components of the model and their possible interactions.

Objective: To mimic the computing potential of mobile Grids considering the effects of fluctuating CPU and limited operation time caused by the finite energy of battery-driven devices. This is in turn done with the aim of advancing research in scheduling mechanisms targeting common performance metrics, e.g. maximize the amount of completed different jobs.

Input: Scheduling criteria relying on static and dynamic parameters of device performance. A list of jobs with varying time of arrival, computing and data requirements. A varying number of devices that act as computing cycles providers, each one associated to traces corresponding to real devices that correlate different levels of computing resources usage (%CPU) with battery depletion for an entire battery discharge cycle. Heterogeneous and independent data transferring cost for each proxy-device wireless link indicated through an Received Signal Strenght Indicator (RSSI) value.

Output: Reports of the scheduling criteria performance and devices behavior, e.g., quantity of completed jobs, job execution time, time in which each job traverse intermediate completion states, devices under utilized periods, devices energy employed by each type resource utilization, etc.

It is worth noting that the representation of the model is subject to the following assumptions and simplifications:

1. Owner and/or OS processes execution cause CPU usage to fluctuate. Grid jobs which run with the lowest priority in a device should adequate their execution to the unused CPU availability.
2. Remaining energy of a device is affected by CPU utilization and networking activity.
3. The computing capability of a device is not modeled at the core level but at the device level. In other words, Grid jobs running in a device are assumed to be programmed so as to use all available cores.

4. The scheduling of mobile Grid computational resources is performed from a centralized node called proxy to which devices maintain a single hop communication link. The proxy is assumed to operate without energetic constraints and within the coverage range of an infrastructure-based communication support, e.g. a WiFi AP or a cellular antenna.

5. Jobs have no data or control dependencies between them (bag-of-tasks Grid applications are modeled), and the execution of an individual job starts and ends in the same device. After finishing executing a job, the device sends the job output data back to the proxy.

4.3.2 The discrete-event driven design of the system model

The conceptual model of Section 4.3.1 gives a panorama of the main concepts and the level of detail that should be present in the logic model. Now, it is presented the object oriented design of the latter and the discrete-event abstraction layer corresponding to the execution engine that runs the configured models. A structural view of the main components are shown in the class diagram of Figure 4.1.

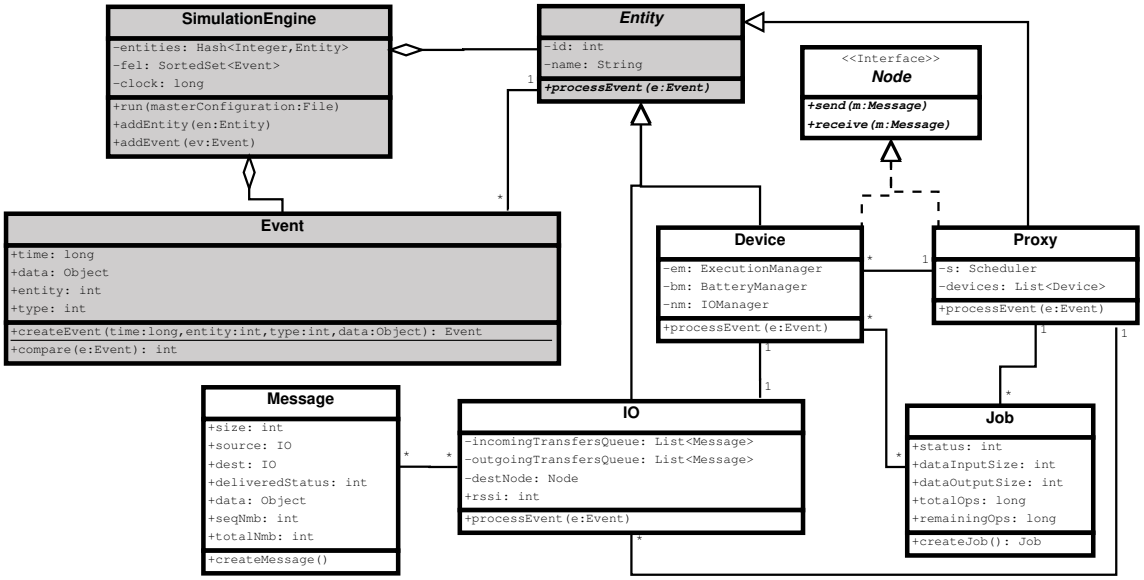


Figure 4.1: Main classes of the infrastructure-based mobile Grid model: UML class diagram

A *Device* object is associated to a *Proxy* object, and this latter is associated to many devices objects. The *Proxy* and *Device* classes implement the *Node* interface that declares basic communication primitives. *Node* is used by *IO* objects. A pair of *IO* objects coordinate the usage of a link and manage the queues of incoming and outgoing data, i.e. they model device-proxy communication. All networking operations between the proxy and a device or vice versa is delegated to such pair of IO objects. Each device has its own IO object while the proxy has many, one per device. The pair of IO instances conforming a point-to-point link has an RSSI attribute that IO objects use to define the delay (and energy spent by battery-driven nodes) of a message transfer operation. The *Message* class represents the fragments in which all data exchanged through a link is split, which is how network protocols operate. The splitting operation is applied to job data input, job data output, devices status updates, and any other future data transfer

supported by the mobile Grid model. By splitting a data transfer into a sequence of messages, the energy consumption of networking operations and those resulting from other utilization sources, e.g., CPU usage, can be interleaved. Modeling this interleaving is important to faithfully reflect energy depletion due to the simultaneous utilization of parallel connected devices components Furthmüller and Waldhorst (2012) (CPU and network card in this case). Lastly, *Job* objects represent the state and attributes of jobs submitted for execution to the mobile Grid system. From the model execution perspective, job and message objects are transient, i.e., objects that flood through different processes which are modeled as entities, e.g., scheduling process, transfer process and execution process.

It is worth to clarify that auxiliary attributes, methods –e.g., setters, getters– and classes have been omitted in the diagram to avoid compromising readability. Among the classes excluded, there are the *ExecutionManager*, *BatteryManager* and *IOManager* classes, which are associated with the *Device* class and encapsulate behavior for managing computing resource availability, remaining battery level and networking related state variables respectively. Moreover, *Proxy* utilizes an *Scheduler* object that contains the logic for deciding which job is assigned to which device. It is important to highlight that the criteria encapsulated by the scheduler object represents a strong point of variability that should be provided by modelers and, as mentioned at the introduction, is what motivates the creation of this tool.

The gray-background classes of Figure 4.1 represent the abstraction layer that encapsulates the type of objects seen by the execution engine. At this point, it is worth to mention that the system dynamics is treated as a discrete-event system. It means that all components (classes) referred in the previous paragraph relate to the classes that encapsulate functioning principles and basic building blocks of such type of simulation. For instance, in Figure 4.1 it can be noticed that *Proxy*, *Device* and *IO* classes inherit from the *Entity* abstract class. An entity object provides a concrete implementation of the *processEvent()* method that receives an *Event* object as argument. An Event instance has four arguments: a time argument that specifies the point in the simulation time where the event occurs, an entity id that gives the engine information of the entity instance/s that should process the event, a type of event argument that gives the entity information of the event nature, and associated data that stores state information or complementary data needed by the entity which processes the event. The events processing could produce system state changes, i.e., changes in the attributes of model components, and/or trigger the creation of new events. Later in this section, it will be described specific events and entities modeled to reflect the system dynamics of the conceptual model. Before that, I will describe the execution protocol utilized to run different infrastructure-based mobile Grid models.

Figure 4.2 outlines the execution protocol –in terms of the involved method call chain– of the simulation engine when the *run* method of the *SimulationEngine* class is invoked. The method firstly initializes and loads all entities and a minimal set of events which are necessary to start the simulation of the model configured by the modeler. Entity instances are maintained in an entity hash, while events instances are added to the *fel* -future event list- of the engine. The initialization consists in reading entities, attributes and events from the *modelConfigFile* provided by the modeler. The whole initialization is performed with several auxiliary classes. There is, for instance, a *SchedulerLoader* in charge of creating the proxy entity. The latter encapsulates the logic/criteria for scheduling incoming jobs. Then, it is invoked the load method of a *DevicesLoader* which, in dual nested loops, creates all devices entities, and for each one

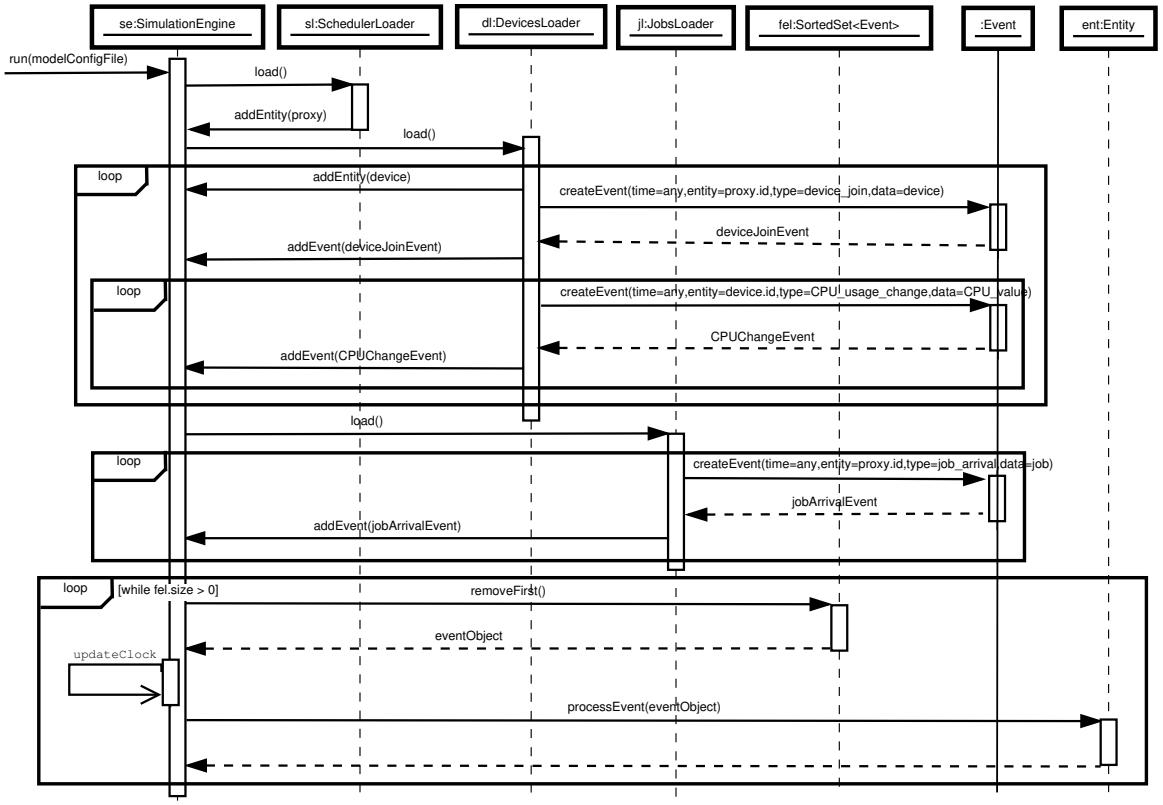


Figure 4.2: Mobile Grid model load and execution: UML sequence diagram

creates and adds a *deviceConnectEvent* and all configured *CPUChangeEvents*. The initialization finishes with the loading of all configured jobs by the *JobsLoader* object. Notice that in every event creation (*createEvent* call of *Event* class) an entity id, which is unique for every entity object, is passed within the arguments of the event constructor method. Such entity id is used to retrieve the entity instance from the list of entities that should handle the event. For example, *CPUChangeEvent* objects have devices id associated, while *DeviceJoinEvents* and *JobArrivalEvents* have the proxy id associated. It worth noting that, during model initialization, the time argument of createEvent calls is, in fact, instantiated with the corresponding time configured by the modeler. For example, if the configured arrival time for $jobId = 1$ is 500 millisecond, then the *time* argument of the createEvent call, that returns the corresponding job arrival event, will be instantiated with value 500.

Before going into details of event scheduling approach that the execution engine follows to simulate the model, it is important to highlight that the initialization phase, which is automatically performed by the engine, is based on the components configuration provided by the modeler. Please refer to Table 4.1 for a list of currently supported configurable attributes for each component. Model variability is mostly supported through configuration files and, as said, the scheduling logic, should be provided programmatically.

Once the initialization phase is completed, the *run* method enters into the model execution loop that consist in repeatedly consuming events from the *fel* sorted set, updating the current clock to the time associated to the consumed event and invoking the *processEvent* method of the entity associated to the event. These steps are repeated until the set is empty. Notice that during a processEvent invocation

new events could be dynamically created and added, which is done by calling the *addEvent* method of *SimulationEngine*. Every event added to the *fel* set is inserted in a chronological order, which is done by comparing events based on their associated time.

Now, I explain how specific components of the infrastructure-based mobile Grid components layer interact through concrete events to reflect the system dynamics. Figure 4.3 gives an schematic overview

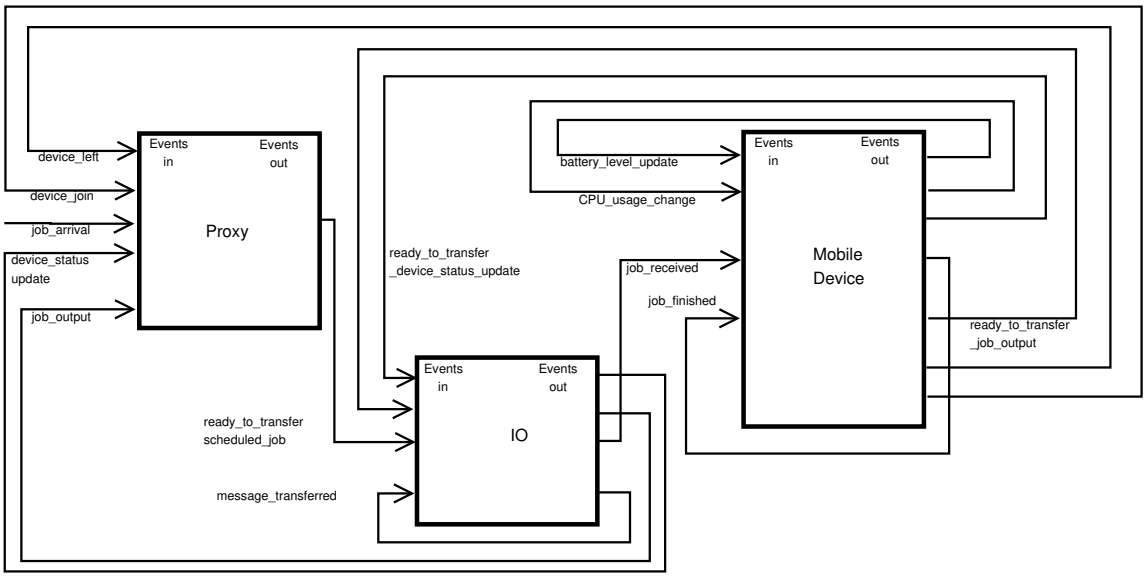


Figure 4.3: Mobile Grid components and events connections: Schematic view with custom notation

of the events processed and created by each component while simulating a mobile Grid. Arrowed lines entering a component -or connecting by its left side- represent input events, i.e., events for which the component provide handling logic. Lines exiting a component by its right side represent output events, i.e., events that the component adds to the simulation engine. Some events are processed by the same component that create them, e.g., battery_update, CPU_usage_change and job_finished are created and processed by a mobile device component. Other events are created by a component but processed by other components: these are any of those departing from the right side of a component and connecting by the left side to a different component. Notice that jobs and messages are not present in Figure 4.3 since they are not modeled as components that process or create events. Instead, jobs and messages are transient objects whose state is updated as they complete a phase in their flow through the system. Job states are differentiated between submitted, input_transferred, finished, and completed, while messages are differentiated between delivered and not delivered. Before describing each component internal functioning, I present Table 4.1, which summarizes the state variables, statistic counters and all currently supported configurable attributes of the model. Configurable attributes can be used by modelers to variate mobile Grid features, e.g., number of nodes, nodes characteristics such as computing capability, energy consumption traces, CPU usage traces, scheduling logic of the proxy, and so on. This is precisely how modelers can exhaustively assess the performance of newly-proposed job scheduling criteria for mobile Grids.

The UML activity diagrams of Figure 4.4, Figure 4.5 and 4.6 outline the actions took by the proxy,

Component	State variables and statistic counters	Configurable attributes
Device	Current battery level, Future battery update, Current job in execution, Current CPU usage, Current owner CPU usage, Queued jobs, Current battery depletion trace, Last notified status	Computing capability (in MFLOPS), initial battery level, baseline CPU usage trace, baseline battery depletion trace, full CPU usage trace, full battery depletion trace
IO	Queued data transfers, Total energy spent in data transferring, data being transferred progress, Current link RSSI, link active	Link RSSI
Proxy	Active connected devices, Submitted jobs	Scheduler criterion, list of connected devices
Message	Sender I/O end, Receiver I/O end, Data type, Data size, Data, Delivered status	Maximum size
Job	Completion status, Execution progress, Device in charge of execution	MFLOP, data input size, data output size, arrival time

Table 4.1: Infrastructure-based mobile Grid simulation model: Components, state variables and configurable attributes

IO and devices components when processing input events. The processing of each input event is represented by individual activities, with their own start and end nodes, and named as "Process <event_name> event". The diagrams show the steps executed by the component which trigger state and/or internal data structures changes. From the execution protocol perspective, the processing of an event is the sequence of internal calls that the entity in charge of handling the event performs before returning the control to the simulation engine. Action nodes whose name start with the "Create event" prefix represent output events, i.e., creation and addition of new events to the simulation engine internal event list. Nodes with an upside-down pitchfork symbol indicates call activity nodes, whose name is the name of an activity surrounded by an visual frame.

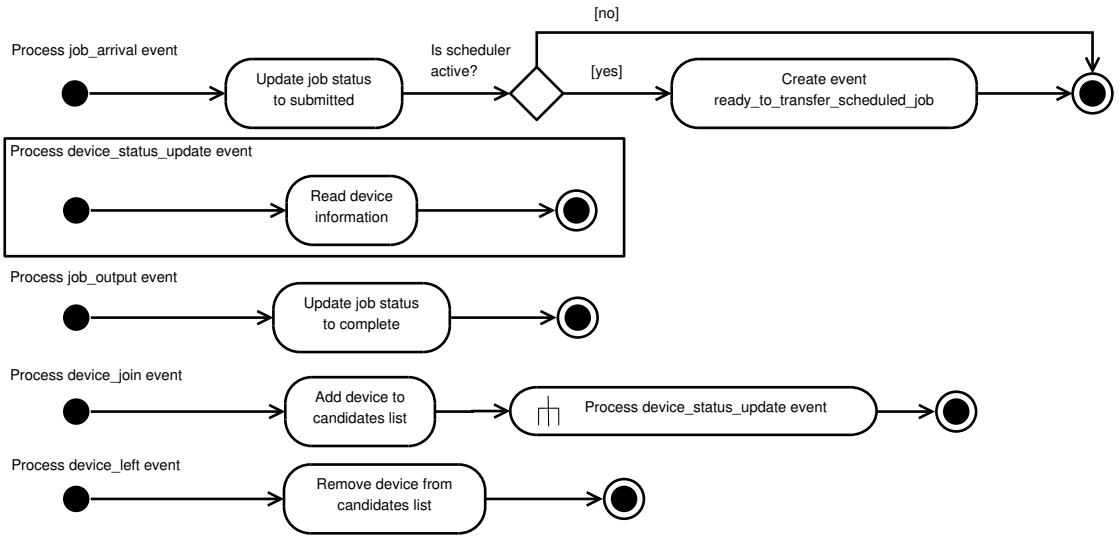


Figure 4.4: *Proxy* events processing and creation: UML activity diagram

When the proxy component processes a *job_arrival* event it updates the status of the associated job object to the submitted status and then evaluates, based on the job scheduling criterion configured by the modeler, whether the job should be immediately assigned (online scheduling) or after some condition is met, e.g., schedule a job bulk size (batch scheduling). Job assignments are represented by the creation of *ready-to-transfer scheduled job* events. The processing of a *device_status_update* event involves that the information of device resources used by the scheduling logic to assign jobs is updated. The event models the lack of update device information managed by the scheduling logic. The processing of a *job_output* event triggers an status update of the associated job which is set to "completed". The processing of a *device_join* event add a device to the list of candidate devices that the scheduling logic can use to assign jobs. After updating the list, the information of device resources is obtained. Lastly, the processing of a *device_left* event involves removing a device from the list of candidate devices to execute jobs.

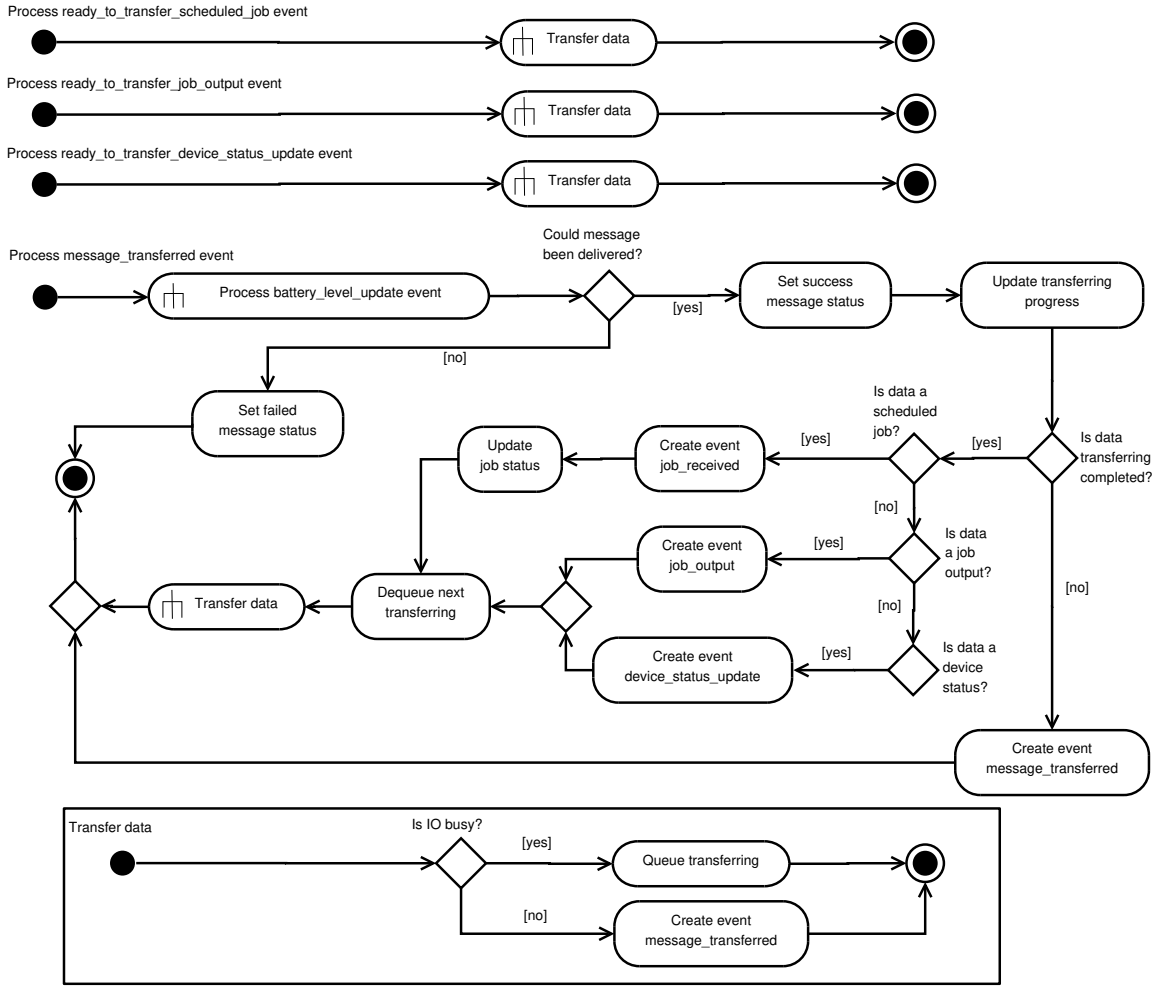


Figure 4.5: *IO* events processing and creation: UML activity diagram

An IO component processes ready-to-transfer and message transferred events. For processing ready_-to_transfer_scheduled_job, ready_to_transfer_job_output and ready_to_transfer_device_status_update events a common sequence of actions is performed. The sequence involves firstly checking whether the IO object is already busy performing other data transfer operation. If there is another operation in

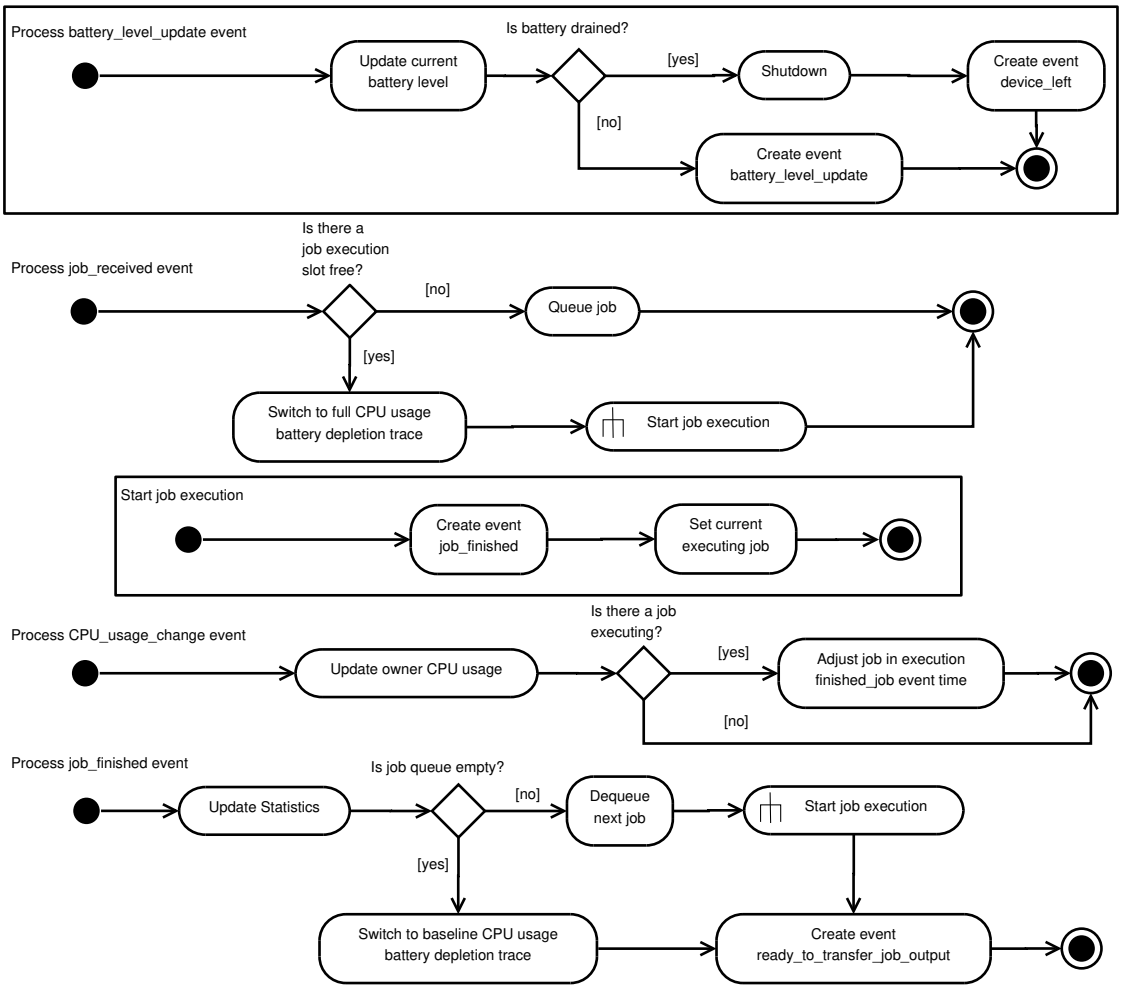


Figure 4.6: *Device* events processing and creation: UML activity diagram

progress, the new data transfer is queued otherwise it starts, which triggers the creation of the first message_transferred event. The time of such event defines the delay after which the IO component of the receiver node processes the message. The current model for sending and receiving messages manages transferring based on a FIFO (First In First Out) queue scheme. Besides, the time of the next message is computed when the time of the previous message is received. The processing of a *message_transferred* event involves to firstly update the current battery level of the node with the corresponding energy consumption associated to the message transfer. Time and energy values are calculated using the message size and the RSSI of the link. Exhaustive empirical studies Ding et al. (2013); Rice and Hay (2010); Balasubramanian et al. (2009) show a correlation between the RSSI indicator and the time spent/energy consumed when transferring data. Then, if the message could not be delivered, i.e., the battery of the node was depleted, the message status is set to "not delivered" and the activity ends. Otherwise, the transferring operation continues by updating the progress of the current transfer. If the transfer is completed, depending on the type of transferring a *job_received* event, *job_output* event or *device_status_update* event is created. The entities associated to those events are Device, the Proxy, and the Proxy, respectively.

With regard to events of the device component, the processing of *battery_level_update* event triggers an

update of the current battery level of the device. After this action is performed, it is checked whether the update do not mean battery depletion. If it is not, the next battery level update event is created, otherwise, a device shutdown is performed. The battery level and time of subsequent battery_level_update events correlate with a certain CPU usage. All this data, in turn, derive from a profiling procedure of real devices, which will be explained in Section 4.4.1. A device shutdown causes that all events to be processed in the future by the device to be removed from the system. Events noticing about job finished, CPU state changes and outgoing messages transfers are included in such remotion. Moreover, the IO object associated to the device is put into an "unreachable" status, which causes delivery failures of all incoming messages to the device. The processing of a *job_received* event starts by checking if any execution slot is available. If not, the number of queued jobs is increased by one. Otherwise, job execution begins, and the following actions are performed:

1. A switch of the current battery depletion trace to the full CPU usage trace occurs. This involves using a set of battery level update events that reflect the energy consumption of a device which is executing a Grid job. Since I is assumed that Grid jobs are programmed to use as much CPU as available in the device, it means that CPU usage elevates to 100%. It does not mean that Grid jobs use the 100% of CPU, since owner processes can be using some percentage of CPU as well. The last does not affect the rate at which energy is consumed but the execution progress of the Grid job.

2. Create a *job_finished* event to occur at time T_j, relative to the current simulation clock. T_j is computed based on the time that the job is expected to be executing in the device, and to know that, the configured number of job MFLOPs and the device available MFLOPS are utilized. The resulting time is the time of the job_finished event.

3. The device variable that points to the current job in execution is updated to reflect the job that has obtained the CPU.

The processing of a *CPU_usage_change* event triggers an update on the device current owner's CPU usage. If there are Grid jobs executing in the device that have not finished yet, the CPU_usage_change event triggers an update of all expected finished times. To make the corresponding update, all current job execution progresses, expressed in terms of remaining FLOP to execute, are calculated by following the equation: $remainingJobMFLOP = totalJobMFLOP - executedMFLOP$ where $executedMFLOP$ are the float point operations of the job executed up to the CPU_usage_change event. After that, job_finished events time are adjusted. The adjustment consists in removing from the main event list the old job_finished events and create new ones with times computed based on the $remainingJobMFLOP$ and the actual value of available device MFLOPS. The available device MFLOPS derive from $nonOwnerCPUusage * deviceMFLOPS$ where $nonOwnerCPUusage = 100 - currentOwnerCPUUsage$. For details about how CPU usage change events representing owner interaction are generated, please refer to Section 4.4.2.

A *job_finished* event triggers an update of the job status to the finished status. The associated device object counter, which maintains the amount of finished jobs, is increased. Besides, if the device queue is

not empty, the next job is dequeued and its execution starts. Otherwise, the device changes to the "idle" state which means that the full CPU usage battery depletion trace is swapped for a baseline CPU usage battery depletion trace. In either case, a ready_to_transfer_job_output event is created to initiate the job output transfer to the proxy.

A key feature of the model is the representation of devices energy consumption derived from CPU usages and IO resources simultaneously. It is appropriate, at this point, to explicit the relation between battery level update events, CPU usage change events, and message transferred events in the determination of the overall device energy consumption.

First recall that, from the mobile Grid perspective, a device state changes from idle to busy -and vice versa-. In idle state, a device is one not executing Grid jobs but not necessarily means that the device is not consuming energy from its battery. In fact, its energy is being consumed at a rate that that is referred as baseline. The data for creating baseline battery level update events derive from samples of the baseline battery depletion trace configured for each device. The time gaps among samples are not arbitrary, but correspond to the rate at which battery level decreases for a targeted CPU usage. This usage, which might fluctuates in time, corresponds to OS and owner processes execution. The data for creating CPU usage change events derived from samples of the baseline CPU usage trace. The usage of traces also applies for a device which is busy, i.e., executing a Grid job. In this case, the energy is consumed at a rate referred as full CPU usage. The data for creating full CPU usage battery level update events derive from samples of the full CPU usage battery depletion trace. A switch between these two battery depletion traces occurs every time a device state passes from idle to busy and vice versa.

Figure 4.7 illustrates an example of the battery level updates (at the top) and CPU usage fluctuation (at the bottom) of a device whose state changes from idle to busy and then to idle again. As the simulation clock advances and the device is idle, its current battery level is updated with samples of the baseline battery depletion trace. At the time a job execution starts, a switch of the battery depletion trace is performed to reflect the device state change from idle to busy. From that time on, the current battery level is updated according to events created with samples information of the full battery depletion trace. When the simulation clock advances to the time in which the the job finished event should be processed (black star), a new switch of the battery depletion trace, in this case, to the baseline battery depletion trace is performed. Upon every switch between battery depletion traces, a synchronization operation is performed by the BatteryManager object of the device to maintain the time-line consistency of the battery depletion. For instance, if at simulation time x, the battery level is 98%, and the device state changes from idle to busy, the synchronization operation involves jump to the sample, of the full usage CPU battery depletion trace, whose battery level matches the current battery level of the device, i.e., 98%. As consequence of the jump, some battery level samples will be discarded as candidate future battery level update events. A similar operation is performed when the device state changes from busy to idle but the synchronization operation is performed over the baseline battery depletion trace.

Resuming the description of Figure 4.7, after the job finished event is processed, the device battery depletion curve shows two scarped slopes. Such battery drops correspond to message_transferred events. At the time a message transferred event is processed by the device IO component, the associated energy cost is discounted from the current battery level of the device. Such operation is performed following the

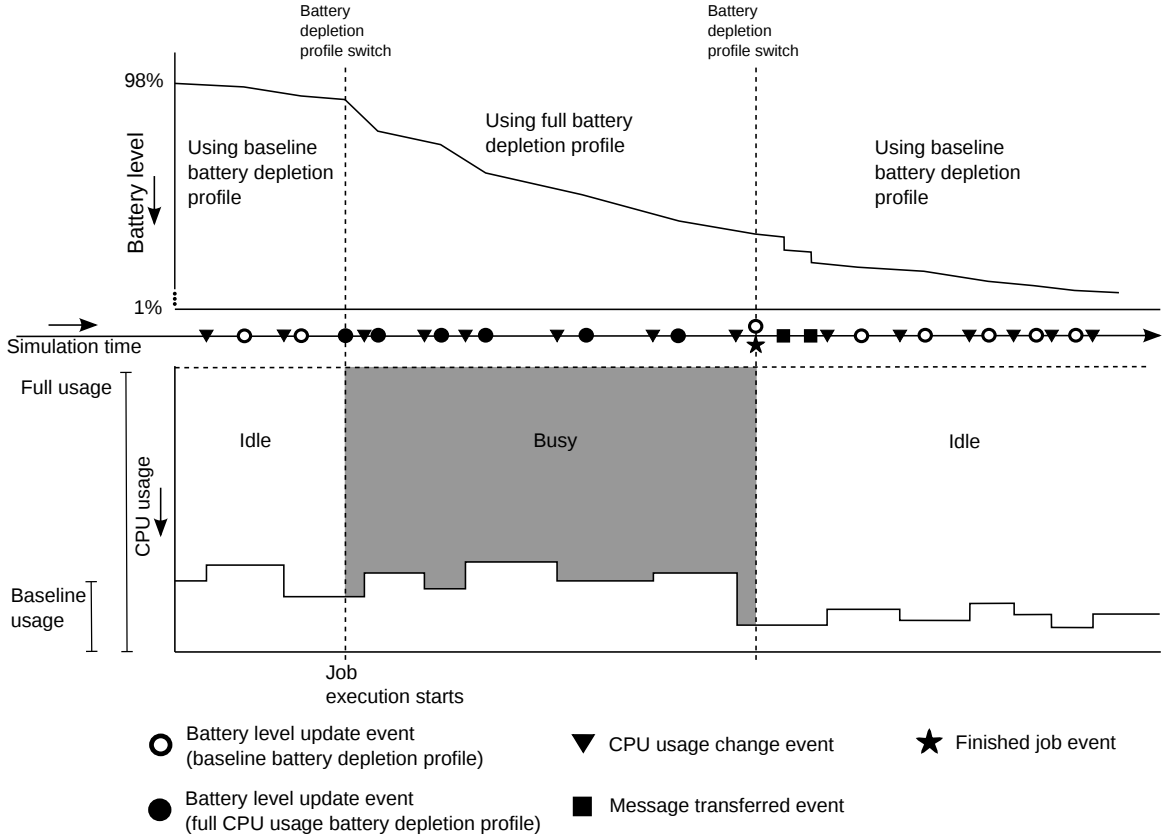


Figure 4.7: Device battery depletion and CPU fluctuation during job execution: Outline of events

next steps:

1. Compute the energy spent by the message transferred event: the message size to be sent or received by the device and the RSSI value of the link are used to compute the energy spent. The model used to compute RSSI values exploits empirical results from a third-party work Ding et al. (2013) studying the energy cost of network usage under different packet sizes and RSSI values.

2. Reflect the message transferred event on the current battery level: since message transferred events can occur at a time t for which there is none battery level update sample associated in a trace, the current battery level at t is obtained by evaluating the linear function that contains the last battery level update and the next battery level update Hoque et al. (2015). The level obtained is decremented by the energy consumption value computed in step 1.

3. Bring future battery level update events backward in time: after the current battery level of the device is updated, the time of future battery level update event needs to be brought backward in time. This operation reflects the shortening of the device lifetime as a consequence of the cumulative energy consumption of parallel connected device components Furthmüller and Waldhorst (2012) (CPU and network card). To perform this operation, it is necessary first to determine the equation of the linear function that joins the last battery level update with the current battery level, and then use the equation to calculate and update the new time in which the next battery level update should occur.

Finally, with regard to the bottom part of Figure 4.7, it is shown that the baseline CPU usage is present while the device is either in the idle or busy states. Baseline CPU usage fluctuation is reflected by CPU usage change events (black triangles). As explained, each CPU usage change events trigger an adjustment of the job finished time currently being executed by the device (not shown in Figure 4.7).

4.4 Simulation input

The configurable attributes of mobile Grid components represent an input source of model. These attributes are specified in easy-to-read configuration files, as explained, parsed by the simulation engine in the initialization phase. There is a configuration file to specify the attributes of each job, including amount of floating-point operations, input and output data size in bytes and arrival time. Arrival time is relative to the simulation start time. Each job specification is converted into a job arrival event, which is added to simulation engine event list. The simulation tool includes a Java application to create job specification files that can be parametrized with ranges of values for each attribute. In turn, attribute values within a range can be generated using probabilistic distributions contains in the `apache.common.math` Java library.

There are also configuration files to specify proxy attributes including scheduling criterion and devices that integrate the mobile Grid, IO and messages attributes, including RSSI for each device-proxy wireless link, and maximum messages size. Other files are used to indicate devices attributes including computing capability, battery capacity in Joules (as indicated by the device manufacturer) and devices usage profiles.

A device usage profile is, in fact, a short name for indicate the files associated to the battery depletion and CPU usage traces. Each device is configured with two usage profiles: a baseline CPU usage profile and a full CPU usage profile. Usage profiles are expected to vary from one device to another, not only to represent different interaction patterns and installed applications but also to represent varying energy consumption as result of varying hardware features. Figure 4.8 illustrate the relation between configuration artifacts -represented by XML and CSV files- with software artifacts of the simulation tool. The execution engine, mobile Grid model and model loader software artifacts have been described in Section 4.3.2. A master or mobile grid model configuration file points to other files that contains the values of the configurable attributes of each component of the model. The figure also shows complementary software, i.e., device profiler, user interaction generator and job generator, available to the modeler for creating variations of mobile Grid components, in summary, different type of devices usage profiles, job lists and schedulers.

Device usage profiles are an important source of variability of devices sub-model. Feeding the simulation with battery depletion events and CPU usage fluctuation events derived from real mobile devices is the core of the in vitro scheme of the simulation tool. Currently, the simulation tool provides a set of ready-to-use usage profiles for the devices models outlined in Table 4.2. The next section explains the procedure for obtaining usage profiles targeting discrete CPU usages for any mobile device. Hence, this procedure can be used to perform simulations including devices other than the ones listed in Table 4.2. Section 4.4.2 deeps into the procedure to interleave several discrete CPU usages profiles to generate new, mixed CPU usage profiles that represent owner's interaction.

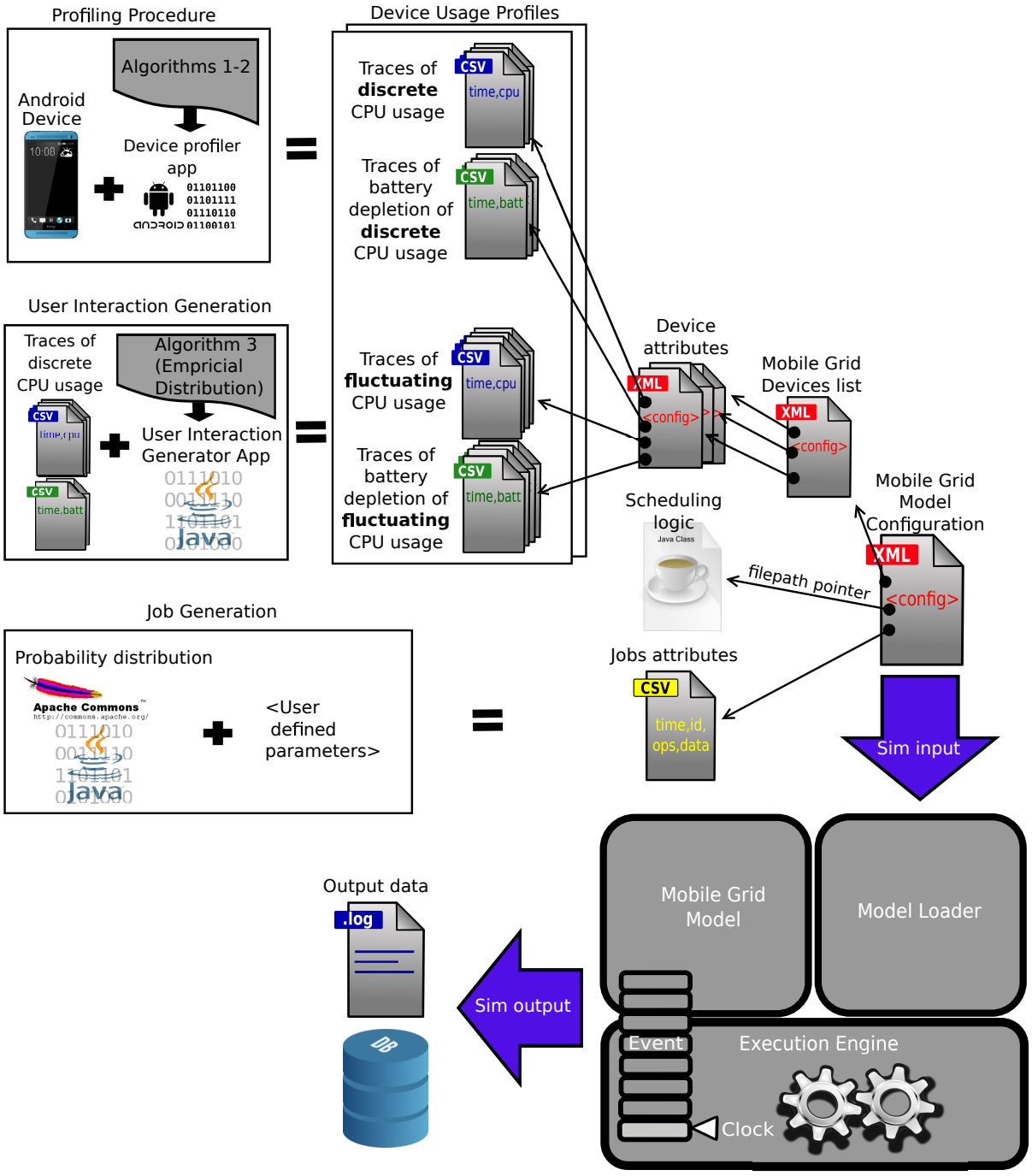


Figure 4.8: Configuration and software artifacts of the simulation tool: Outline

4.4.1 Mobile device profile generator

As explained, data to create battery level update events and CPU usage change events is extracted from traces and a set of traces are used by the device sub-model to represent idle and busy states. The traces are obtained from profiling real devices. Device profiling is an off-line procedure performed with an Android application, but the proposed profiling methodology can be easily adapted to other mobile device platforms, such as iOS or Windows Mobile.

The *device profiler* app, from now on simply device profiler, monitors three variables, *time, CPU usage* and *battery charge*, and keeps CPU usage near to the selected CPU usage target. Time is measured, using the mobile device internal clock. CPU usage is measured by a dedicated thread that uses the Linux file `/proc/stats` for reading the current CPU state (as the `top` Linux command does) and reads the file every 200 milliseconds. The CPU used is calculated by averaging the last 30 measurements to reduce noise. *Battery charge* is measured using the event based system battery report via the Android Intent API Mednieks et al. (2012). The device profiler logs samples of battery charge and the time, upon each new battery event issued by Android system, in a battery depletion trace file. At the same time, time and CPU usage changes samples are logged in a CPU usage trace file.

The device profiler operates by performing floating-point operations in separate threads (one thread per core). Operations are executed in chunks separated by a sleeping time. Each thread consumes nearly 100% of each core when the sleeping time is zero. Therefore, a monitoring component periodically adjusts a delay time between the operations to support lower *target CPU usage*.

Algorithm 4.1 shows the CPU usage generator algorithm run per core. The sleeping time is adjusted according to Algorithm 4.2, which runs in another thread. In contrast, the number of operations (CYCLES) is fixed. This constant had to be defined because current mobile devices processors are relatively fast, so it was impossible to control the CPU usage by only executing few floating-point operations and sleeping. In all the usage samples built so far (see Table 4.2), CYCLES equals one million. Yet, this number should be re-configured for other processors.

Algorithm 4.1 aims at artificially generating CPU usage by performing floating-point operations. The SLEEP parameter regulates how much CPU is used. When SLEEP equals 0, the algorithm generates near 100% CPU usage. To adjust SLEEP for other CPU usage targets, Algorithm 4.2 runs in another thread, modifying SLEEP according to the rate between current CPU usage and target CPU usage. Finally, when target CPU usage is 0%, neither algorithm is run.

Figure 4.9 shows a screen-shot of the device profiler GUI. The software should be initiated on a mobile device with fully charged battery and plugged to the electrical power line. When pushing the “Start” button, the software begins to generate CPU usage attempting to find an sleep time that results into a CPU usage near to the *target CPU usage* within the given threshold. Upon convergence, a notification indicates the user to unplug the device and left the software run without human interaction until battery depletion. The usage profile is stored on the device SD card. During the profiling procedure, WiFi should be ensured to stay active since a mobile device offering resources to an infrastructure-based mobile Grid is expected to be connected to a wireless network. Then, the energy consumed in keeping the WiFi on is considered, which produces more realistic usage profiles.

```
1:  procedure CPUUsageGenerator
2:      while true do
3:          if SLEEP > 0 then
4:              wait(SLEEP)
5:          end if
6:          count ← 0
7:          while count < CYCLES do
8:              performFloatingPointOps
9:              count ← count + 1
10:         end while
11:     end while
12: end procedure
```

Algoritmo 4.1: CPU usage generator algorithm

```
1:  procedure CPUUsageAdjuster(TargetCPUUsage, Threshold)
2:      while true do
3:          cpuUsage ← getCPUUsage
4:          diff ← cpuUsage/TargetCPUUsage
5:          if −Threshold < 1 − diff < Threshold then
6:              NotifyStable                    ▷ CPU usage is near TargetCPUUsage
7:              Log(cpuUsage)
8:          else
9:              sleep ← CPUUsage.getSleep()
10:             if sleep = 0 then
11:                 sleep ← 1
12:             end if
13:             CPUUsage.setSleep(sleep ∗ diff)
14:         end if
15:     end while
16: end procedure
```

Algoritmo 4.2: CPU usage adjuster algorithm

Note that the Android scheduling subsystem is very aggressive and might terminate the software, which intensively uses the CPU, especially when the mobile device screen is locked. To avoid this, the threads run within an Android Service in foreground. A Service Mednieks et al. (2012) is an application component that performs long-running operations without graphical interface. Another issue is that Android might reduce the CPU frequency to preserve battery. However, it is assumed that CPU of a device is fully used when it executes a job of the mobile Grid. Android provides *power locks*, which allow user applications to tell the OS whether the applications need to keep some part of the device fully active. In particular, *partial wake locks* Mednieks et al. (2012) are exploited to keep the CPU active, but not the screen or the physical key lights on.

4.4.2 The user modeling module

A baseline profile provides information of the actual non-available CPU, when the device is busy. As explained in Section 4.3, the time of a *job_finished* event is computed based on it. When a device is idle, the non-available CPU information is not used. Instead, the useful information the baseline profile is the battery depletion data associated to the CPU usage.

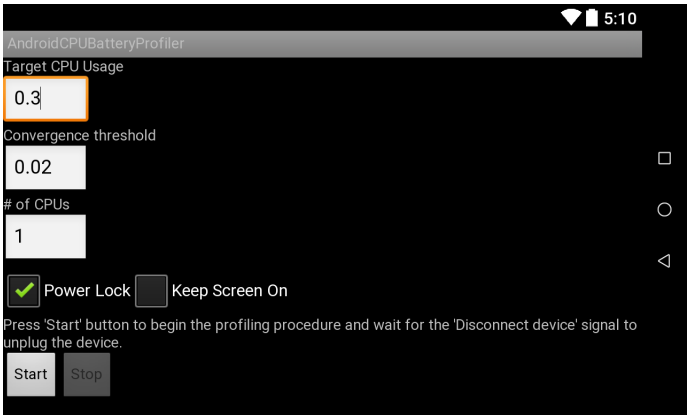


Figure 4.9: Device profiler app: Screenshot

Baseline profiles targeting discrete CPU usage for the whole device battery discharging cycle might not represent a real user interaction. For this reason, it is also provided a methodology to build baseline profiles that imitate CPU fluctuation (and the corresponding energy consumption) caused by processes and applications as a result of owner's interaction. To this end, discrete CPU usage profiles i.e., 0%, 30%, 75% and 100%, generated via the methodology described in Section 4.4.1 are interleaved. Interleaving is not random but employs empirical probabilistic models of a third-party study Falaki et al. (2010) to obtain two key parameters that describe user interaction, namely user session length and time between sessions. The study analyzed intentional user interactions over a dataset of 33 Android users monitored over 9 weeks in average and 222 Windows Mobile users monitored over 16 weeks in average. It is worth to clarify that the baseline profile which results from this interleaving procedure does not include energy consumption related to device data transferring that can be involved with the owner interaction other than Grid jobs data, which is a future work.

The probabilistic models capture three aspects of device usage (interaction sessions, interaction time and diurnal patterns). First, it is exploited the sub-model describing *interaction sessions*, i.e., session lengths or interaction intervals, which is given by:

$$SessionLength = r * Exp(\lambda) + (1 - r) * Pareto(x_m, \alpha) \tag{4.1}$$

where r is the relative mix of the two distributions, λ is the rate of the Exponential distribution, and x_m and α are the location and shape parameters of the Pareto distribution. In Falaki et al. (2010) it is concluded that most sessions are very short and the frequency drops exponentially as the session length increases. But, inconsistently with the exponential behavior, there are some very long sessions in the tail for each device user, which are modeled with Pareto. Moreover, it is exploited the sub-model associated to *interaction time*, i.e., the elapsed time between a session and the next one by each user. The sub-model uses a Weibull distribution together with scale and shape parameters.

These two sub-models are further refined by using a discretization of the curves that authors inferred for each distribution parameters (r, λ, x_m, scale/shape for Weibull). Furthermore, I selected a value of $\alpha = 0.2$ since this is the average value found for a 60% of the use cases via a percentile analysis Falaki et al. (2010).

```
1: procedure CREATEMIXEDPROFILE(cpu0Profile, profilesWithCpuUsage[ ],
   sessionLengthModel, sessionTimeModel)
2:     mixedProfile ← empty
3:     currentBatteryPercentage ← 100%
4:     usageProfileIndex ← 0
5:     upperLimit ← 0
6:     profileType ← 0                          ▷ 0/1 for including events of 0%/above 0% of CPU
7:     while currentBatteryPercentage >= 1% do
8:         if profileType == 0 then
9:             offset ← sessionTimeModel.GENERATESAMPLE
10:            events ← GETEVENTS(cpu0Profile, upperLimit, offset)   ▷ Events from upperLimit plus
   offset
11:            APPEND(mixedProfile, events)          ▷ events are appended to mixedProfile events
12:        else
13:            offset ← sessionLengthModel.GENERATESAMPLE
14:            events ← GETEVENTS(profilesWithCpuUsage[usageProfileIndex], upperLimit,
   offset)
15:            APPEND(mixedProfile, events)
16:            usageProfileIndex ← usageProfileIndex + 1 mod profilesWithCpuUsage.SIZE
17:        end if
18:        currentBatteryPercentage ← mixedProfile.GETCURRENTBATTERYPERCENTAGE
19:        upperLimit ← offset
20:        profileType ← profileType + 1 mod 2
21:    end while
22:    return mixedProfile
23: end procedure
```

Algoritmo 4.3: User interaction-driven base profile generator algorithm

Samples of the above sub-models are used to define the time when a new interleaving action should be introduced (time between session) and the duration of the CPU usage profile employed for the interleaving action (session length). As suggested above, interleaving includes discrete CPU usage profiles of 0% (to simulate standby state), 30% (to simulate an interaction requiring low-mid CPU demand), 75% (to simulate interactions requiring mid-high CPU demand) and 100% (to simulate interactions requiring high CPU demand). This allows modelers to model infrastructure-based mobile Grids with varying owner-device interaction profiles (e.g., office users versus gamer users).

Algorithm 4.3 shows the procedure used to interleave profiles. As input, the procedure uses a 0% CPU usage profile, a profiles array of constant CPU usages, a session length generator logic and a time between session generator logic based on the above probabilistic sub-models. Figure 4.10 shows examples of usage profiles representing CPU usage fluctuation caused by user interaction for two tablets (Acer A100 and Viewsonic ViewPad 10s).

It is important to highlight that the interleaving procedure should not be performed at runtime, i.e. while running the mobile Grid model. One of the reasons is the reduction of the effective simulation time. Interleaving discrete CPU usages during simulation would heavily increase the computational overhead and memory requirements of mobile Grid model simulations. This is because the interleaving procedure involves loading into memory and sequentially read several files with tens of thousand battery and CPU samples and a relatively small fraction of all those files samples are finally included in the baseline profile. Besides, from all those included, many of them can be discarded during simulation. Computational

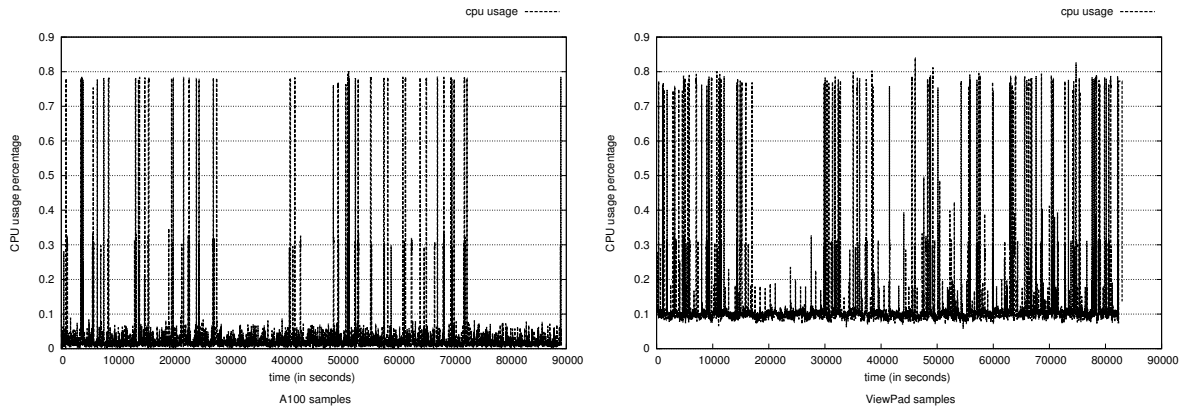


Figure 4.10: Examples of base usage profiles with fluctuating CPU usage derived from user interaction

overhead could be overcome with powerful hardware, however, irreproducibility of experimental conditions would not be accepted in any case. This is the other reason why the interleaving procedure should not be performed at runtime. By sampling information from a probability distribution, the samples values and order that give shape to the experimental conditions, cannot be assured for several runs which is important when several scheduling strategies are compared.

4.5 Simulation output

Running a mobile Grid model outputs a log of status changes experimented by devices and jobs. Examples are device x joining/leaving the mobile Grid, job a assigned to device x, and job a finishing in x. Status changes have a time stamp, which is useful to derive scheduling metrics within a time window. For instance, by filtering all executed jobs, *throughput* –percentage of executed jobs– can be computed. Additionally, by tracing back the status changes of a job, it is possible to determine the device(s) which handled its execution due to job re-assignations, or counting how many jobs were successfully executed by specific devices (e.g., those having over m FLOPS).

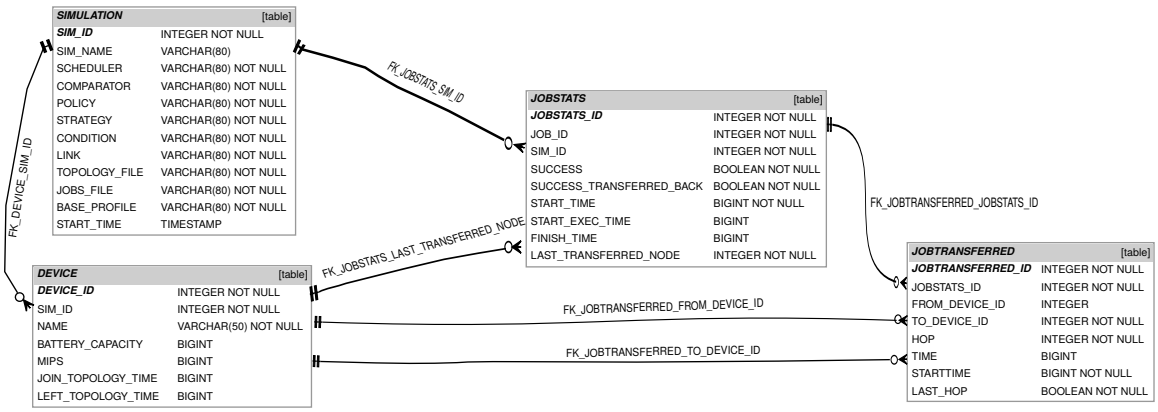


Figure 4.11: Simulation output: Entity-Relation Diagram

For each job, the simulator logs its identifier, arrival time at the proxy, and status changes ("submitted", "input_transferred", "finished", and "completed"). For each device, the simulator logs its identifier, the

time at which it joins/leaves the mobile Grid, when it receives/starts/finishes a job, and when it changes to idle state.

These data along with extra performance metrics are logged to standard output. The simulator has classes that allow the output to be persisted in a relational database, whose entity-relation diagram is shown in Figure 4.11.Simulation stores the parameters for a mobile Grid model instance, including the Java class that implement the scheduling criteria, the files for configuring jobs and devices attributes, among others. A simulation is associated to tuples in Device, which stores the devices information. Moreover, `jobstats` stores jobs data. Each job tuple is, in turn, associated to tuples in `jobstransferred`, which stores jobs transfers related information, i.e.,times when input/output transfer is initiated and terminated, and the total/current distance in hops w.r.t. the device that originally offloaded a job. This latter is since jobs can be re-scheduled between devices several times until it is actually executed by a specific device.

4.6 Validation

At the core of the infrastructure-based mobile Grid model is the logic to reflect battery consumption of mobile devices due to executing Grids jobs, owner/OS processes, and sending/receiving data through the network. As explained in past sections, the time between battery level updates have a correspondence with a CPU usage, and this information is obtained from traces that are unique for a specific device brand. Moreover, energy consumed in transfer operations is computed and cumulatively discounted from the device current battery level. Recall that battery level is updated with data extracted from the baseline and the full usage CPU profiles. Jumps between both usage profiles are synchronously done for mimics the battery depletion caused by energy-hungry activities happening in a device.

The simulator assumes that the charge at any time between two samples in a trace can be approximated using a linear function. This is since mobile device OSs report battery level changes at discrete steps, and then, the real discharge function between two reports are unknown. However, the energy model is not built upon the assumption that the whole set of discharge samples are adjusted to a single linear function. The reasons behind this decision are validated in Section 4.6.1. On the other hand, in Section 4.6.2 there are presented experiments to validate that the above scheme suffices to account for the extra energy consumed when transfer data in addition to using CPU. It is also shown that using baseline usage profiles with fluctuating CPU usage also serves to account for the effects of user's interaction in energy availability and task termination times.

4.6.1 Discharge model validation

In this experiment, 24 profiles where generated using 6 different Android devices under 4 discrete CPU usages. Since these profiles contain discrete battery charge samples only, it is assumed that the charge at any time between two samples in a trace can be approximated using a linear function. This is because energy consumption at any given moment maintaining the same CPU usage is expected to remain con-

stant. However, it does not mean that this linear behavior is profile-wide, which is a strong assumption upon which the device battery depletion sub-model discussed in Section 4.3 builds.

Device	Type	Target CPU usage in %	Generated CPU usage in % (mean)	Generated CPU usage (standard deviation)	Number of samples	Elapsed time (hh:mm:ss)
ViewSonic ViewPad 10s	Tablet	0	10.23	2.26	5,476	27:15:39
		30	30.11	2.02	4,154	19:29:27
		75	76.93	1.79	2,951	13:49:49
		100	99.98	0.07	2,424	11:20:24
Samsung I5500	Smartphone	0	3.83	1.01	7,030	35:23:08
		30	30.06	1.44	3,466	15:37:12
		75	75.62	1.04	2,174	11:14:22
		100	99.98	0.04	2,065	09:45:25
Acer A100	Tablet	0	2.03	1.49	4,877	27:08:35
		30	30.34	3.07	2,139	10:15:13
		75	77.78	1.02	1,810	8:28:06
		100	99.98	0.11	1,557	7:17:01
Samsung Galaxy SIII	Smartphone	0	3.11	2.44	14,752	74:35:37
		30	30.18	3.15	10,162	48:17:27
		75	75.78	1.38	3,855	18:06:29
		100	99.98	0.31	1,332	6:13:37
Samsung Galaxy Tab 2	Tablet	0	1.24	1.26	29,520	147:44:32
		30	30.17	1.79	12,065	56:45:10
		75	76.14	1.36	5,060	23:43:48
		100	99.74	1.07	4,185	19:37:18
LG Optimus l9	Smartphone	0	0.69	0.56	9,261	46:16:42
		30	30.25	2.85	4,634	21:47:42
		75	76.13	1.32	1,227	5:44:25
		100	99.99	0.11	932	4:21:24

Table 4.2: CPU usage generated with the profiler app

Regarding the CPU usage generated by the device profiler app, Table 4.2 depicts the average CPU usage generated and its standard deviation. In all cases, the standard deviation was less than 4%. The worst trace was generated for the Viewpad 10s with a target CPU usage of 0%. However, in this case the profiler does not generate CPU usage, so the CPU usage was mostly generated by Android OS and pre-installed applications. Visually, the table shows that each trace has a fairly constant CPU usage regardless the target CPU usage and the device.

Device	Type	Target CPU usage in %	Slope	Avg. slope between data points	Standard deviation	MAE	MAE Ranking
ViewSonic ViewPad 10s	Tablet	0	-9.6649e-07	-1.0036e-06	1.5613e-07	0.9230	7
		30	-1.3403e-06	-1.3984e-06	4.2052e-07	0.9702	8
		75	-2.1044e-06	-2.0661e-06	1.1525e-07	0.6557	5
		100	-2.4676e-06	-2.6582e-06	1.9584e-06	1.0460	10
Samsung I5500	Smartphone	0	-6.1921e-07	-1.3731e-06	2.6462e-06	7.3122	23
		30	-9.6927e-07	-1.8618e-06	4.2646e-06	6.2197	22
		75	-2.0773e-06	-2.9023e-06	2.7395e-06	5.1535	21
		100	-4.0704e-06	-6.9098e-06	2.1892e-05	7.6921	24
Acer A100	Tablet	0	-9.7457e-07	-9.7457e-07	4.8063e-07	1.1511	13
		30	-2.6036e-06	-2.6984e-06	2.4494e-07	1.3690	15
		75	-3.1817e-06	-3.2294e-06	2.6793e-07	0.8034	6
		100	-3.7897e-06	-3.9345e-06	3.6718e-07	1.3207	14
Samsung Galaxy SIII	Smartphone	0	-3.6936e-07	-4.1044e-07	1.1489e-07	2.7278	19
		30	-5.7783e-07	-6.2874e-07	1.6604e-07	2.4341	18
		75	-1.5268e-06	-1.6278e-06	3.0953e-07	2.1199	16
		100	-4.2384e-06	-4.5708e-06	7.0395e-07	2.3279	17
Samsung Galaxy Tab 2	Tablet	0	-2.0572e-07	-2.3048e-07	7.4680e-08	2.8096	20
		30	-5.0410e-07	-5.0403e-07	1.4248e-08	0.1275	1
		75	-1.1739e-06	-1.1795e-06	2.5609e-08	0.2745	2
		100	-1.4090e-06	-1.4200e-06	3.4190e-08	0.3181	3
LG Optimus l9	Smartphone	0	-6.2271e-07	-6.3943e-07	2.0427e-07	1.0620	11
		30	-1.2884e-06	-1.3445e-06	2.9517e-07	1.0677	12
		75	-4.9520e-06	-5.1157e-06	1.0162e-06	1.0262	9
		100	-6.5652e-06	-6.5879e-06	1.1522e-06	0.5740	4

Table 4.3: Linear regression: Battery charge given time

To empirically assess whether there is a linear relationship between time and battery, it is performed a linear regression minimizing the Mean Squared Error (MSE) by gradient descent. The goal was to provide a charge estimation (C_e) given a time (T), and an initial charge (IC) using Equation 4.2 that minimize the error (E) as defined in Equation 4.3, where C_i is the real charge for each logged battery sample i, EC_i is the estimated charge for that sample and N is the number of logged samples. Here, IC is known (100) because the battery is fully charged at $T = 0$. Hence, the only parameter to adjust is the slope (S).

$$C_e = S.T + IC \tag{4.2}$$

$$E = \frac{\sum_{i=1}^{N}(C_i - EC_i)^2}{N} \tag{4.3}$$

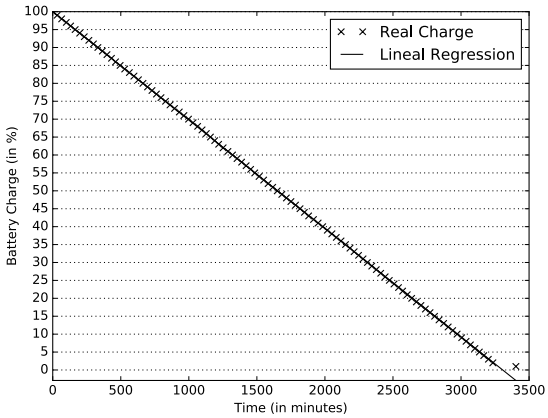
Table 4.3 depicts the slopes obtained for the different traces considering the charge (0 to 100). This table also presents the average slope between two arbitrary data points and its standard deviation. Finally, it also shows the Mean Absolute Error (MAE), as defined in Equation 4.4, for the profile and its standard deviation:

$$E = \frac{\sum_{i=1}^{N}|C_i - EC_i|}{N} \tag{4.4}$$

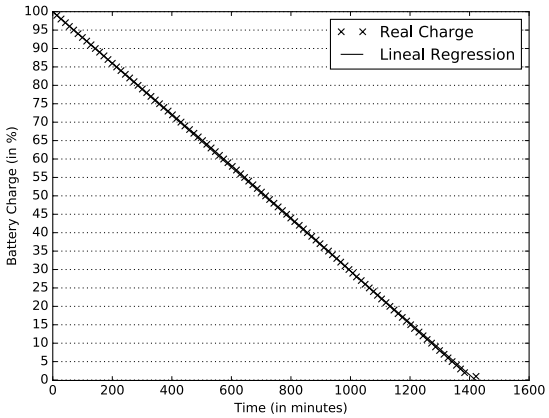
In short, in some cases, there seems to be a linear relationship between time and charge: Figure 4.12 depicts both real charge and estimated charge for the three traces which obtained the lowest MAE. There is no noticeable difference between the regression and the real data. Furthermore, Figure shows 4.13 the same information, but for the linear regressions with highest MAE. It can be seen that the data points are not far from the estimated charge, which supports the idea that when mobile device CPU usage is rather constant a linear function provides an accurate estimation Hoque et al. (2015). However, for many profiles, MAE values are not negligible. This is the base of the device battery depletion model, which instead of performing a profile-wide adjustment to a linear function, it only assumes linearity between two trace samples.

4.6.2 Network usage and user's interaction validation

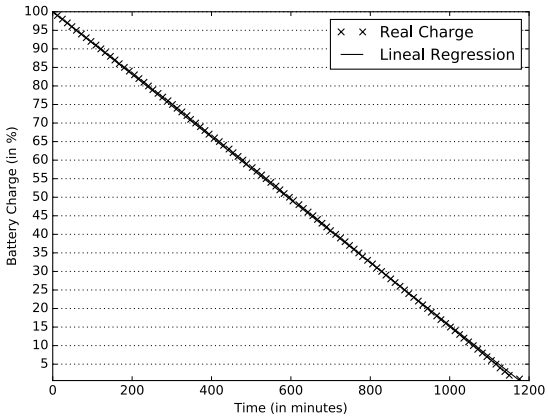
Figure 4.14 depicts validation tests to show how the mobile Grid model accounts for the effects of user interaction and data transferring over jobs and energy consumption. To show the user interaction effect, i.e., CPU usage and energy shared among user processes and Grid jobs, different mobile Grid settings are simulated by varying the amount of dedicated devices (those without user activity). All mobile Grid settings are composed by one hundred devices –20 Acer A100 devices, 41 ViewSonic Viewpad 10s



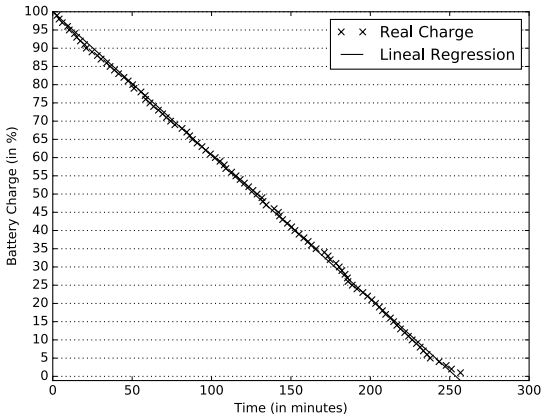
(a) Galaxy Tab 2 - 30%



(b) Galaxy Tab 2 - 75%

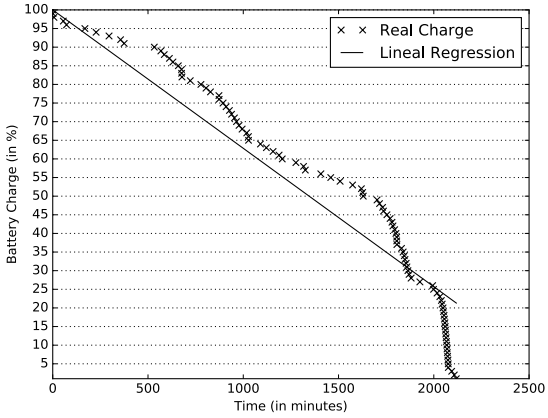


(c) Galaxy Tab 2 - 100%

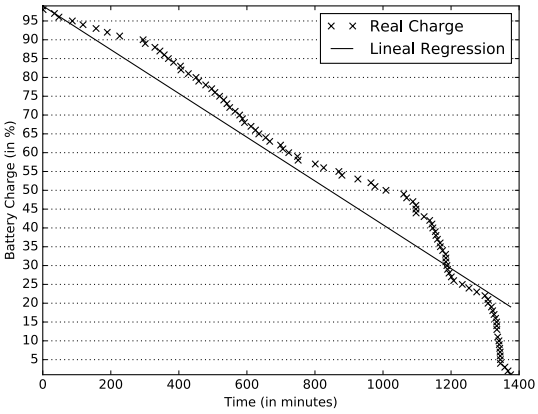


(d) LG Optimus L9 - 100%

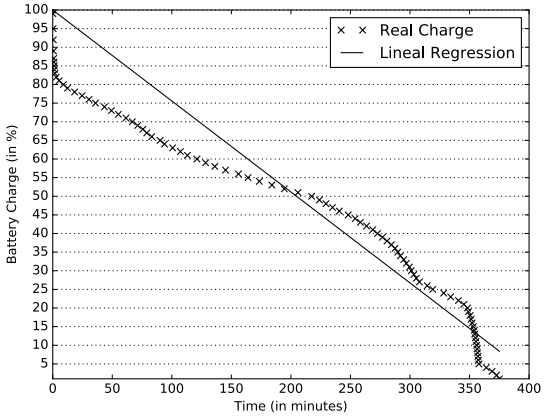
Figure 4.12: MAE: Four of the best ranked profiles



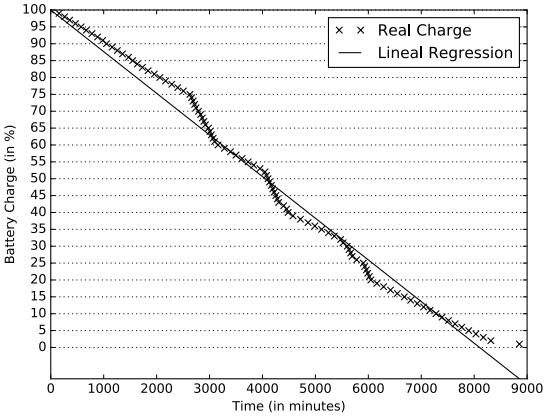
(a) I5500 - 0%



(b) I5500 - 30%

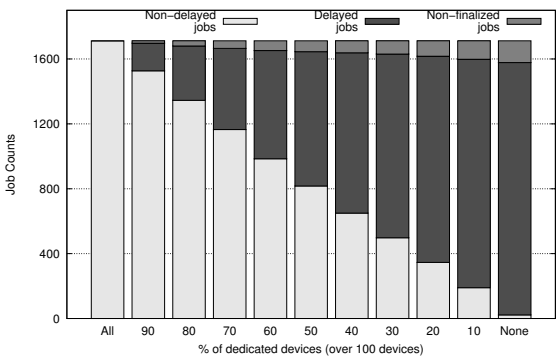


(c) I500 - 100%

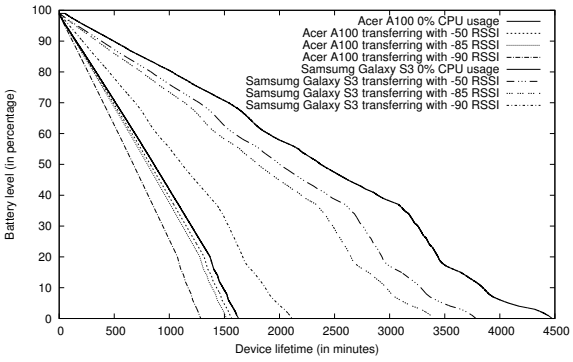


(d) Galaxy Tab 2 - 0%

Figure 4.13: MAE: Four of the worst ranked profiles



(a) CPU dedication effect on jobs turnaround time and jobs state



(b) Battery depletion effect of data transfers at different RSSI for two device brands

Figure 4.14: User interaction and data transferring validations

devices and 39 Samsung I5500 devices– and process the same jobs set. Job assignments are performed with a scheduling criterion that assigns as many consecutive arrived jobs to each device as the latter is able to execute. To this end, the heuristic uses the number of FLOP each job requires to be executed and the maximum number of FLOPs each device is able to execute with its available energy. Figure 4.14a depicts how jobs state changes w.r.t. a reference mobile Grid setting with all devices its dedicated. The more the devices with non-dedicated baseline profiles are included in a mobile Grid setting, the more the job executions are delayed, and the more job executions do not finalized, which shows that CPU usage caused by user interaction do not only alters the finish time of Grid jobs but also the devices energy used to perform them.

To validate the mechanism that cumulatively accounts for the energy spent on data transferring using a battery depletion traces referring to CPU usage, it is performed a test that consists in making a device to transfer one megabyte at intervals of two minutes, and compare its lifetime degradation for different transferring costs. To this end, it is simulated a mobile Grid setting with several devices of the same device brand configured with the same baseline battery depletion profile (0% CPU usage) but connected to the proxy with different RSSI values that directly lead to different transferring costs. All devices are initialized with the same battery level (full-charged battery) and the transferring test is repeated until their batteries are depleted. Figure 4.14b depicts how lifetime of two specific device brands varies considering different RSSI values. The Samsung Galaxy S3 smartphone lifetime is more heavily affected than the Acer A100 table lifetime. This is because the smartphone is equipped with a battery with less capacity in Joules than that of the tablet. The figure also explains why the cost associated to the same data transferring operation at the same RSSI -which is simplified to be the same for all devices brands-, represents a bigger battery percentage drop in the smartphone compared to the tablet.

Chapter 5

Experimental evaluation

5.1 Experimental scenarios

In this section, it is delineated the design of experimental scenarios used to validate the work hypotheses stated in Section 3.2. All scenarios run in a controlled simulated environment whose functioning principles, technical details, and scope were explained in Chapter 4. Heterogeneity is reflected in experimental scenarios through different variables including SMDs computing capability, composition of MVR instances, jobs sizes and job arrival rates. Below, I explain the meaning and justify the selection of these variables, as well as, the values considered.

SMD computing capability refers to the throughput an individual node is able to deliver. This can be measured in the traditional form, i.e., as a rate of float-point operations performed by the node within a time window, or as a rate of amount of completed work per energy unit. Throughput of fixed computers is usually associated to computing hardware features, however, in SMDs throughput also implies battery-related features. From this perspective, SMDs equipped with different computing and battery hardware, such as the Acer Iconia Tab A100 tablet, the ViewSonic ViewPad 10s tablet and the Samsung I5500 smartphone, render different throughputs (see Table 3.1). Given such differences, these three SMD models are selected as representative of heterogeneous computing capabilities for the designed scenarios.

Apart from hardware features, other source of heterogeneity that characterizes SMD computing capability is owner interaction. As explained in Chapter 4, owner interaction profiles make the available computing capability to fluctuate in time and the energy not to be consumed in processing instructions of external jobs. Recall from Section 4.3.1 that external jobs are assumed to be programmed to use as much CPU as available in an SMD. It means that the computing capability of an SMD available to execute external jobs is subject to how intensively in CPU terms the owner interacts with the device. See Section 4.4.2 for details of simulated interaction profiles.

From the resource availability perspective, computing capability of SMDs is classified into *dedicated* and *non-dedicated*. A dedicated SMD model means that its computing resources are exclusively employed to execute external jobs. By contrast, when it is non-dedicated computing resources (including energy) are shared to execute external jobs and owner processes.

Given the variability of SMD models that might coexist in a resource exploitation opportunity, other aspect present in the design of scenarios is the heterogeneity of MVR composition. The MVR instances defined for the experimental scenarios have the particularity of being composed by the same number of SMDs (100) and similar aggregated computing capacity (ACC). Such a decision was made to favor cross-MVR comparisons of finalized jobs. MVR instances, however, differ in the quantity of each SMD model and total Joules with which the ACC is provided. The ACC of a MVR is expressed as the sum of all SMDs individual computing capacity ($ICC_{SMDmodel}$), which, in turn, is defined as:

$$ICC_{SMDmodel} = MFLOPS_{SMDmodel} * DischargeBatteryCycle_{SMDmodel} \tag{5.1}$$

where $MFLOPS_{SMDmodel}$ denotes the Mega Float point Operations the SMD model is able to perform per second, and $DischargeBatteryCycle_{SMDmodel}$ the time, assuming a full-charged battery, the SMD model lasts running at 100% of CPU usage. Then, the ICC values of the SMD models used in the experimental scenarios are 1.6E^6, 0.85E^6 and 0.266E^6 for Acer Iconia Tab A100, ViewSonic ViewPad 10s and Samsung I5500 respectively. To create MVR instances with similar ACC supported by the same number of SMDs the following equation is used:

$$ACC = 1.6E^6 * X + 0.85E^6 * Y + 0.266E^6 * Z \pm \varepsilon \tag{5.2}$$

, where X, Y and Z are the resulting quantity of A100, ViewPad 10s and I5500 nodes respectively.

Variable	Value	Description
SMD Comp. capability	Dedicated	without owner interaction (owner CPU usage of 0%)
	Non-dedicated	owner CPU usage fluctuates between 0%, 30% and 75%
MVR instance	MVR1	Nodes combination: 10 A100, 64 ViewPad 10s, 26 I5500 ACC: 77.320 TFLOP. Energy: 6,445,008 Joules
	MVR2	Nodes combination: 20 A100, 41 ViewPad 10s, 39 I5500 ACC: 77.220 TFLOP. Energy: 5,035,752 Joules
	MVR3	Nodes combination: 30 A100, 18 ViewPad 10s, 52 I5500 ACC: 77.132 TFLOP; Energy: 3,626,496 Joules
Jobs set	Short	1,910 jobs with size in the range [3,232.27 - 9,696.82] MFLOP
	Long	11,608 jobs with size in the range [19,393.65 - 59,180.95] MFLOP
Job arrival rate	$150x10^{10}$ FLOTES	Job arrival time window of 50 seconds
	$12.5x10^{10}$ FLOTES	Job arrival time window of 10 minutes
	$6.25x10^{10}$ FLOTES	Job arrival time window of 20 minutes

Table 5.1: Variables and values of the simulated scenarios

When creating MVR instances targeting the same ACC value but with the constraints that total number of nodes, i.e. $X + Y + Z$, should be the same and X, Y and Z should be positive integers, an error ϵ

appears. That error, for the MVR instances created, is ±188,000 MFLOP, which represents a difference of approximately 0.24% in computing capacity. Table 5.1 outlines the values of X, Y and Z for the three MVR instances created along with their ACC values and total number of Joules.

Moreover, when evaluating the performance of schedulers it is natural to consider heterogeneity in the sets of external jobs. In my experiments, heterogeneity is present in job sizes, which are expressed in Million Float point Operations (MFLOP), and in the time each one arrives to the proxy. With regard to job sizes, it is considered a set of long jobs with operations in the range [19,393.65 - 59,180.95] MFLOP, and a set of short jobs with operations that vary in the range [3,232.27 - 9,696.82] MFLOP. The long and short jobs sets are generated by sampling size values from a continuous uniform distribution, so that to assure a balanced number of job of different sizes. The base of this decision is to ensure more heterogeneous set of jobs than sampling sizes, for instance, from a Normal distribution. The number of long jobs and short jobs in each set are 1,910 and 11,608, respectively. These values are selected according to the ACC of MVR instances. The idea is to exceed the MVR ACC, so as to avoid reaching a hundred percent of finalized jobs, which would hinder the comparison of the schedulers performance. All jobs have a fixed input/output data size. Input represents data and code that need to be present at the node in charge of executing a job, while output represents the job result transferred to the proxy. Input and output data sizes do not represent a variable in simulated scenarios, and are set to 1,024 bytes and 4 bytes respectively for all jobs. Those numbers are reasonable for CPU-bound jobs, which is the type of targeted applications of the proposed schedulers.

In online scheduling, it is also appropriate to consider the time between jobs arrivals as a variable, specially because the criteria for distributing jobs changes with information generated while the system runs, e.g., periodic battery updates sent by SMDs. Considering that jobs are described in terms of amount of FLOPs, a unified way of indicating job arrivals, irrespective of jobs quantity and size, is through Float-point Operations To be Executed per Second (FLOTES). The indicator defines the rate of Float point operations that the system receives for execution within a time window. It is worth to clarify that several values of this variable are only studied for the set of experiments which validates the hypothesis 3.2 -first phase experiments-. The reason behind not extending the study of several values of the variable to the experiments which validate hypothesis 3.2 and hypothesis 3.2 -second phase experiments- is because these two hypothesis refer to the re-balancing phase of the approach where runtime information is expected to be already generated. More specifically, at the time the re-balancing phase is triggered, all jobs are expected to be already distributed to SMDs, several of them might have finalized and consequently the corresponding updated information is expected to be available for re-balancing. The FLOTES values utilized are $150x10^{10}$, $12.5x10^{10}$ and $6.25x10^{10}$, meaning that all jobs, either short or long, arrive within a time window of 50 seconds, 10 minutes and 20 minutes, respectively. For the experiments of the second phase jobs arrive at a rate of $150x10^{10}$ FLOTES. Table 5.1 summarizes the variables and values described along this section.

5.2 Simulation results

In this section, it is reported and analyzed the performance of the two-phase scheduling approach, instantiated for the different energy-aware criteria and job stealing techniques proposed in Chapter 3. The performance is measured in number of finalized jobs, because the more the jobs a scheduler finalizes, the more energy-efficient it is. The terms "device" and "node" will be used interchangeably to refer to SMD instances.

The results are organized as follows: Section 5.2.1 deeps into details about the first allocation phase performance for all energy-aware criteria proposed. The results derived from the simulated scenarios and the tests which prove statistical significance of results are used to validate hypothesis 3.2. It is appropriate at this point to clarify that the Round Robin scheduler is employed as the representative strategy of non energy-aware schedulers against which the energy-aware criteria proposed for the first phase are compared. The Round Robin scheduler has been selected due to several reasons. One of these is that, like all our battery-aware criteria proposed, it is designed with practical applicability in mind since it does not depend on jobs requirement information to operate. The other reason is that it is a widely known strategy in the Distributed Computing community that has been used as baseline for assessing scheduler performance.

Section 5.2.2 details the performance achieved by the second allocation phase, i.e., the gains of the rebalancing techniques with regard to the performance achieved by the first allocation phase. The results of these scenarios and the tests proving the statistical significance of results are used to validate the hypothesis 3.2 and hypothesis 3.2. Finally, Section 5.2.2.2 presents a discussion about the energy implications of the two-phase scheduling instantiations which best performed in average.

5.2.1 First phase: Evaluation of ranking-based battery-aware criteria

The first allocation phase is the part of the decision making process that assigns a job as soon as it arrives to the proxy. The results reported in this section show the performance that non battery-aware and battery-aware schedulers achieved for the scenarios designed. The Round Robin (RRN) is the representative of non battery-aware schedulers, while the E-SEAS, JEC and FWC are the battery-aware schedulers. The results are used to validate the hypothesis 3.2, which states that using battery-related information in the SMDs ranking allows the system to achieve better throughput, i.e., to finalize more jobs.

Table 5.2 summarizes the performance ranges, measured in number of finalized jobs, that each scheduler achieved for all simulated scenarios of the first allocation phase. Each range comprises the results of dedicated and non-dedicated scenarios considering all job arrival rates outlined in Table 5.1. The analysis of results is supported with the histograms of Figure 5.1, Figure 5.2 and Figure 5.3. The histograms area are split in three vertical parts and each part corresponds to the performance of schedulers using different MVR instances. The area associated to an MVR instance is, in turn, split to differentiate dedicated and non-dedicated scenarios. The latter correspond to those bars with gray-out background. It is worth to mention that the analysis of schedulers performance made for short jobs scenarios also applies for long

Job Set	MVR	Non battery-aware	Battery-aware		
		RRN	E-SEAS	JEC	FWC
Short	MVR1	[9,429 - 9,605]	[10,026 - 10,147]	[10,130 - 10,422]	[9,432 - 9,949]
	MVR2	[8,477 - 8,614]	[9,776 - 10,047]	[9,851 - 10,070]	[8,522 - 9,280]
	MVR3	[7,506 - 7,625]	[10,159 - 10,481]	[9,472 - 9,684]	[7,519 - 8,595]
Long	MVR1	[1,504 - 1,528]	[1,619 - 1,647]	[1,625 - 1,674]	[1,499 - 1,573]
	MVR2	[1,346 - 1,370]	[1,577 - 1,629]	[1,577 - 1,616]	[1,355 - 1,428]
	MVR3	[1,194 - 1,213]	[1,637 - 1,699]	[1,520 - 1,567]	[1,195 - 1,313]

Table 5.2: First phase finalized jobs of non battery-aware and battery-aware schedulers

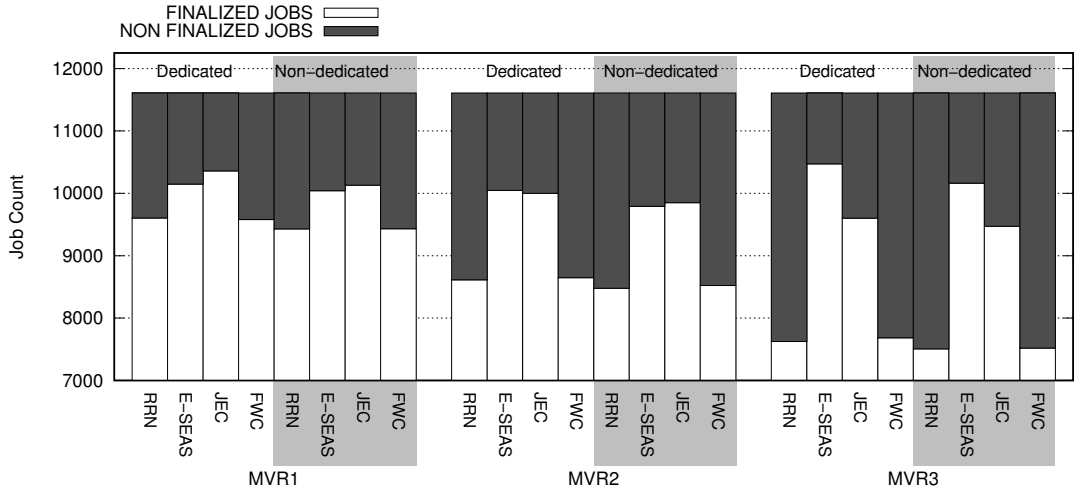
job scenarios, unless it is specifically indicated. To ease the interpretation of schedulers performance, I will also express the count of finalized jobs in percentages over the size of the corresponding job set.

Figure 5.1 outlines the performance of all schedulers considering a job arrival rate of $150x10^{10}$ FLOTES, which means that all job assignments are made within the first 50 seconds, i.e. the time interval in which the whole set of jobs arrive to the proxy. Figure 5.1a and Figure 5.1b discriminates the performance of all schedulers for short and long job scenarios, respectively. Figure 5.1a shows that, in MVR1 scenarios, E-SEAS and JEC finalize notably more jobs than RRN. FWC performance is not as high as the other battery-aware schedulers. In fact, its performance is quite similar to RRN. E-SEAS outperforms RRN by [4.67 - 5.28]%. JEC improvements are within the range [6.04 - 6.49]% and FWC difference is in the range [(-0.22) - 0.03]%.

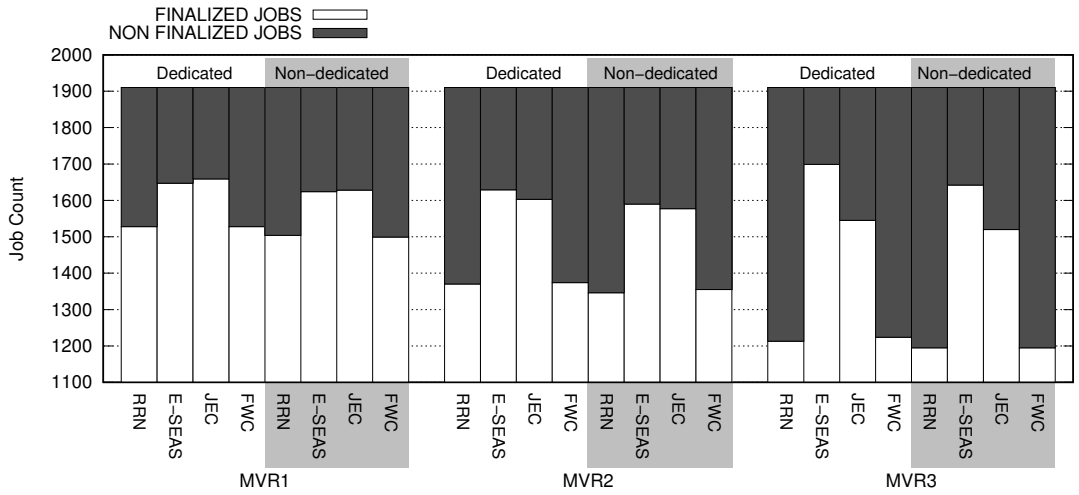
In MVR2 scenarios, the differences in favor to battery-aware schedulers increase. E-SEAS, for example, outperforms RRN in [11.31 - 12.34]%, JEC in [11.84 - 11.93]%, and FWC in [0.28 - 0.39]%. In MVR3 scenarios, improvements in favor to battery-aware schedulers were also obtained. These improvements are [22.88 - 24.53]%, [16.94 - 17.03]% and [0.11 - 0.5] for E-SEAS, JEC and FWC respectively. A similar behavior, with approximately the same magnitude of differences, are registered for all schedulers in long job scenarios.

The incremental pattern of improvements observed with a cross-MVR analysis seems to be related to MVR computing capability. To be more specific, as the computing capability of mid-end SMDs, i.e., ViewPad nodes, is switch by that of high and low-end SMDs, i.e., A100 and I5500 nodes respectively, the battery-aware schedulers performance increases compared to RRN.

The 50 seconds-window in which all jobs arrive to the proxy, whose results were previously described, represent a stressed operation condition for the schedulers. This is since neither battery nor jobs information is updated in such a short time interval. The performance of all schedulers is further analyzed for two slower job arrival rates, which represent less stressed conditions for all criteria and favor quicker update of ranking formula components built upon dynamic information. Figure 5.2 shows the performance considering a job arrival rate of $12.5x10^{10}$ FLOTES, which means that all jobs arrive -and are scheduled- within the first 10 minutes of simulation. Such a time interval represents less stressed conditions for all



(a) First phase finalized jobs of all schedulers for short job scenarios



(b) First phase finalized jobs of all schedulers for long job scenarios

Figure 5.1: First phase performance of all schedulers using a job arrival rate of $150x10^{10}$FLOTES

schedulers than a time interval of 50 seconds. In less than 10 minutes all devices are able to sent at least one battery update message, which causes their respective ranking to vary at the proxy. Such updates are specially capitalized by the FWC scheduler whose ranking criteria is purely based on dynamic information. The improvements w.r.t. RRN for short job scenarios are in the range of [1.76 - 2.21]% in MVR1 scenarios, [4.88 - 5.2]% in MVR2 scenarios and [7.93 - 8.22]% in MVR3 scenarios. When comparing the FWC performance in short jobs scenarios for a job arrival rate of $12.5x10^{10}$ FLOTES (Figure 5.2a) with the performance achieved by the same scheduler for job arrival rate of $150x10^{10}$ FLOTES, it is observed a gradual take off effect represented by increments in the ranges of [1.73 - 2.42]%, [4.49 - 4.91]% and [7.71 - 7.82]% for MVR1, MVR2 and MVR3 respectively.

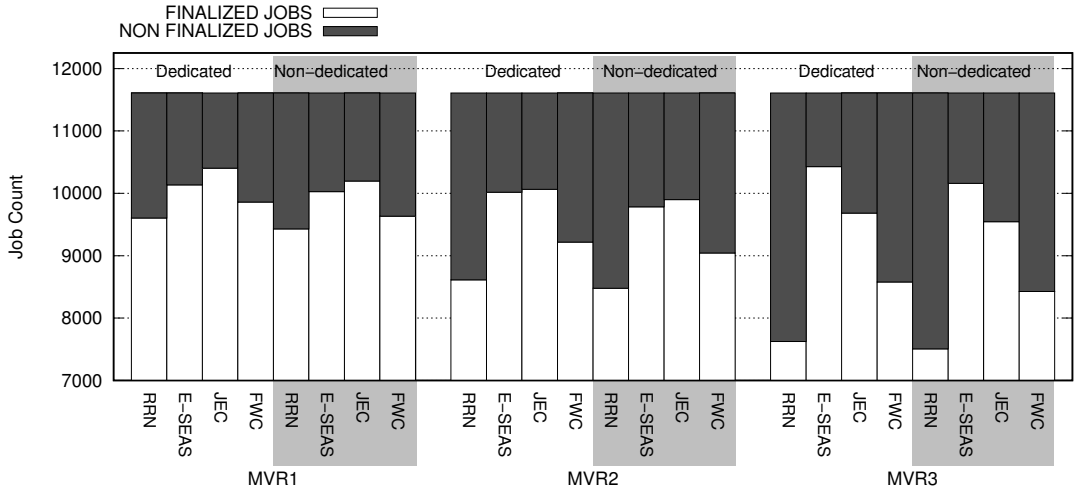
When the same analysis is applied to E-SEAS and JEC, i.e., their own performance is compared w.r.t. their performance achieved for scenarios of $150x10^{10}$ FLOTES, there are observed small performance decrements for E-SEAS and small increments for JEC. E-SEAS decrements are in the ranges [(-0.14) - (-0.09)]%, [(-0.26) - (-0.06)]% and [(-0.37) - (-0.03)]% for MVR1, MVR2 and MVR3 respectively. By contrast, JEC increments are in the ranges [0.4 - 0.58]%, [0.4 - 0.55]% and [0.64 - 0.71]% for MVR1, MVR2 and MVR3 scenarios respectively. Despite these small performance differences, both battery-aware schedulers still clearly outperform RRN.

In long jobs scenarios with $12.5x10^{10}$ FLOTES (Figure 5.2b) FWC scheduler performance also improves w.r.t. to that of RRN. Improvements are in the ranges [0.63 - 1.2]%, [1.57 - 1.78]% and [1.78 - 2.62]% for MVR1, MVR2 and MVR3 respectively.

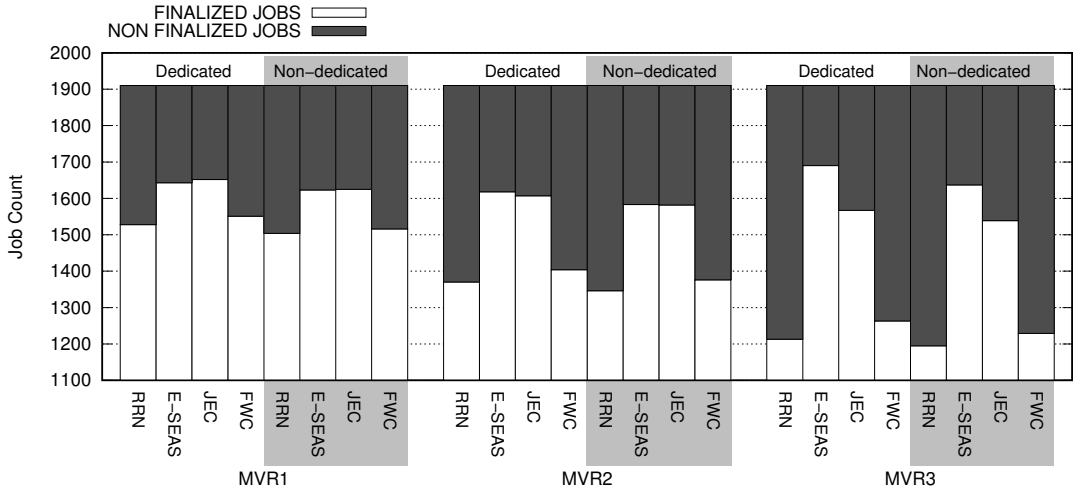
Figure 5.3 outlines the performance of all schedulers for a job arrival rate of $6.25x10^{10}$ FLOTES, which equals to a time interval of 20 minutes. In this time interval, more battery updates are received and potentially alter the rank of devices at the time scheduling decisions are made. The results reveal that neither for short jobs (see Figure 5.3a) nor in long jobs (see Figure 5.3b), E-SEAS and JEC loose competitiveness w.r.t. the non battery-aware scheduler. The same affirmation applies to FWC, whose performance is always better than that of RRN. Now, in respect to the incremental improvement of FWC along different job arrival rates, in short job scenarios, it seems that the criterion experiences a convergence effect, which is not experienced for long job scenarios. For the latter, new improvements are obtained in the ranges [1.83 - 2.36]%, [3.04 - 3.19]% and [4.66 - 5.24]% for MVR1, MVR2 and MVR3 respectively.

5.2.1.1 First phase concluding remarks

After comparing the performance of the three battery-aware schedulers with that of a representative non battery-aware scheduler with controlled experiments for heterogeneous conditions represented by scenarios with different job sizes, SMDs computing capability, MVR composition, and job arrival rates, there is evidence to accept the statement of hypothesis 3.2. The statistical significance of such results is, in turn, tested with two-side Wilcoxon tests. The test is appropriate to compare matched samples of two populations, to assess whether the mean of such populations differ. The samples matching is given by the performance of a non battery-aware scheduler for a specific scenario with that of a battery-aware scheduler for the same scenario. It means that independent tests are run for all non battery-aware

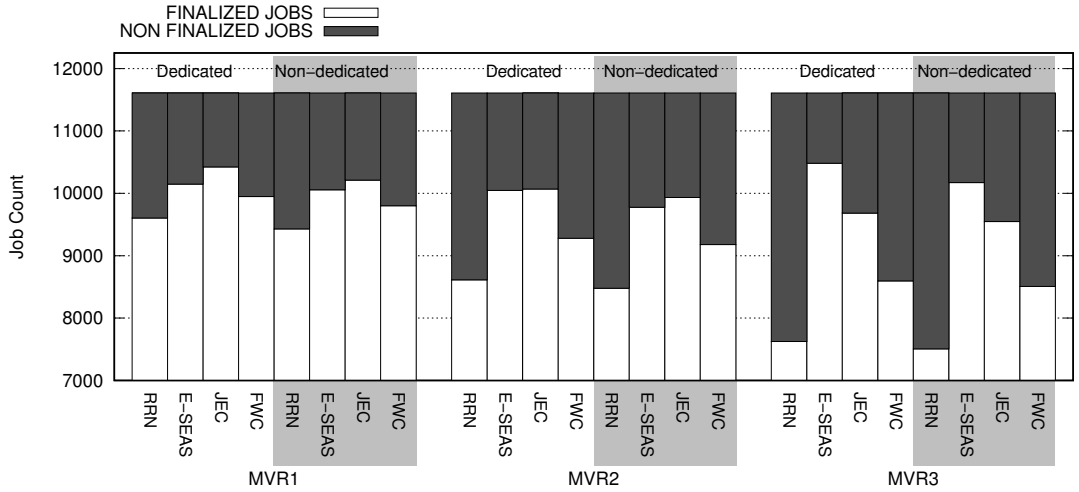


(a) First phase finalized jobs of all schedulers for short job scenarios

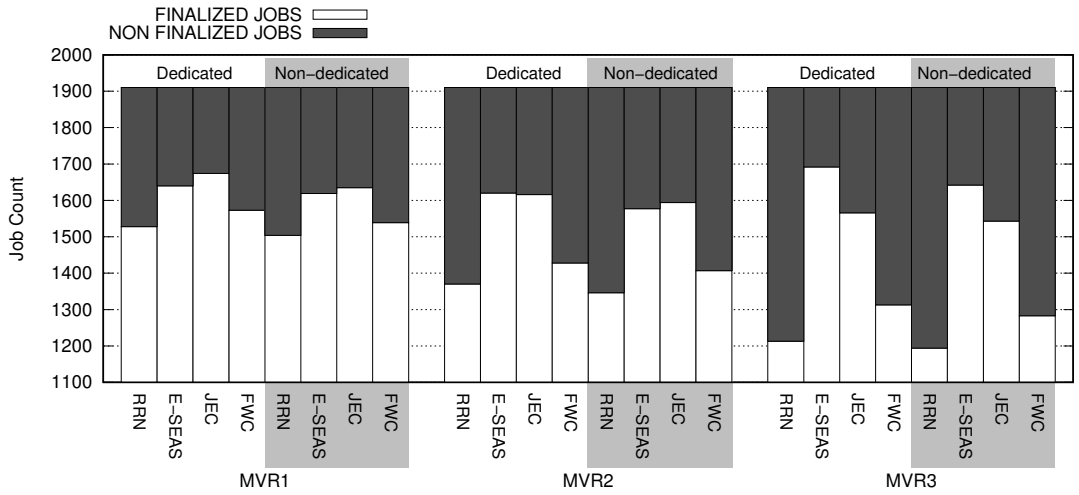


(b) First phase finalized jobs of all schedulers for long job scenarios

Figure 5.2: First phase scheduling criteria performance for a job arrival rate of $12.5x10^{10}$FLOTES



(a) First phase finalized jobs of all schedulers for short job scenarios



(b) First phase finalized jobs of all schedulers for long job scenarios

Figure 5.3: First phase scheduling criteria performance for a job arrival rate of $6.25x10^{10}$FLOTES

- battery-aware combinations. The data sets of each test are the performance values, i.e., number of finalized jobs for all scenarios.

The null hypothesis (H_0) is formulated as follows: the mean number of finalized jobs achieved with a non battery-aware scheduling criterion is equal to the mean number of finalized jobs achieved by a battery-aware scheduling criterion. The alternative hypothesis (H_1), then, states that the mean number of finalized jobs achieved with a non battery-aware scheduling criterion differs from the mean number of finalized jobs achieved by a battery-aware scheduling criterion.

Non battery-aware scheduler	Battery-aware scheduler	P-value
RRN	E-SEAS	0.000000167809576708
RRN	JEC	0.000000167882883144
RRN	FWC	0.000000634258559543

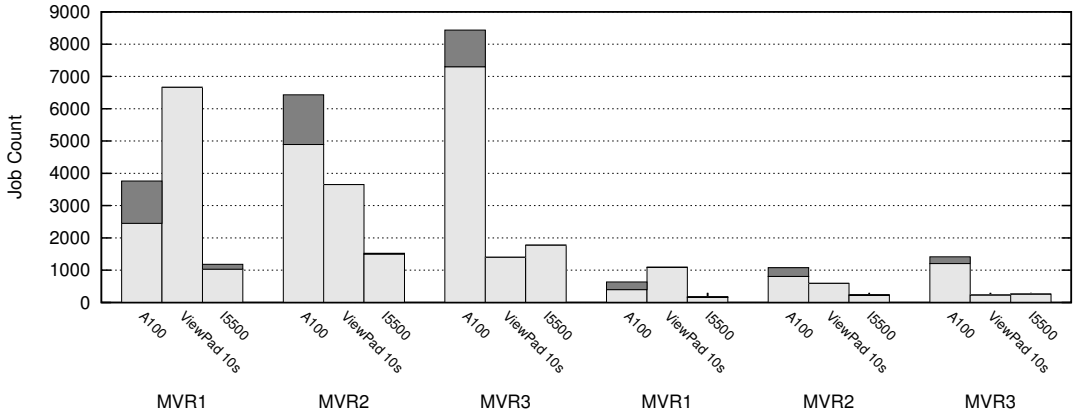
Table 5.3: P-values of statistical tests for non battery-aware vs. battery-aware schedulers

The p-values resulting from two-side Wilcoxon tests are outlined in Table 5.3, and indicate that H_0 is rejected for all non battery-aware - battery-aware scheduling combinations with a significance level of $\alpha = 0.01$. Having evidence that the mean number of finalized jobs of all battery-aware criteria is over the mean number of finalized jobs of the non-battery aware criterion, and these means are statistical significant, the hypothesis 3.2 is accepted.

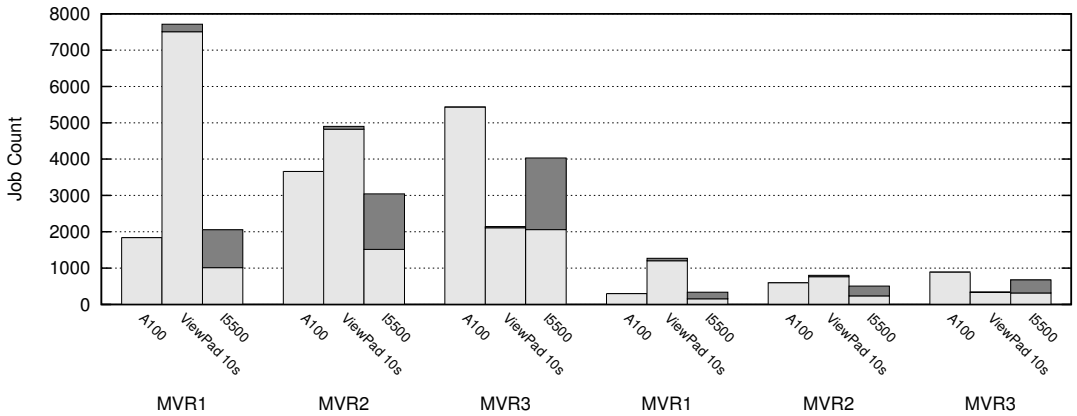
5.2.1.2 First phase battery-aware schedulers observations

The following are observations of the battery-aware schedulers behavior that could originate further work hypotheses and are made based on the scenarios run and presented in Section 5.2.1. One of such observations is that battery-aware schedulers which achieve the best performance alternates between JEC and E-SEAS, while FWC is always relegated to the third place. Moreover, a cross-MVR analysis shows that the number of finalized jobs by JEC and FWC schedulers degrades as more A100 devices are included in the MVR instance. By contrast, the performance of the E-SEAS scheduler slightly increases with such inclusion, or at least, the performance seems to be less affected than the other two battery-aware schedulers.

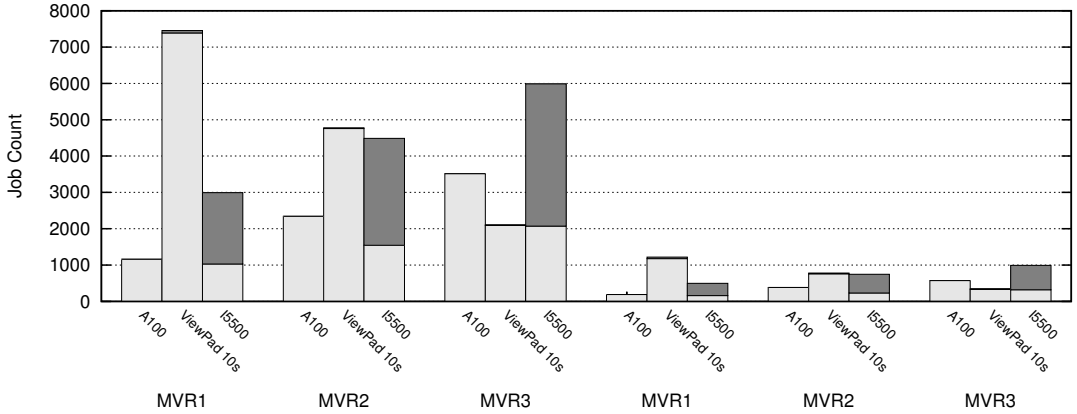
A zoom into the jobs final state, shown in Figure 5.4, provides insights of schedulers operation for different node types. Bars on the left half correspond to short job scenarios while bars on the right to long job scenarios. Non-finalized jobs are indicated in dark gray while finalized jobs in light gray. The following analysis is done for dedicated scenarios but also applies for non-dedicated scenarios. In Figure 5.4a it is observed, for all MVRs and job sets, that non-finalized jobs are mainly accumulated in A100 devices and a comparatively small quantity in I5500 devices. By counting with the computing capability of Viewpad nodes instead of that of Acer and I5500 nodes, which is progressively given in MVR1, MVR2 and MVR3, the performance of the criterion is not penalized. This seems to be related to the fact that E-SEAS nodes ranking prioritized exploitation of these type of nodes more than exploitation



(a) E-SEAS short job and long job scenarios



(b) JEC short job and long job scenarios



(c) FWC short job and long job scenarios

Figure 5.4: Jobs final state: Occurrences discriminated by node type

of average nodes (Viewpad) when balancing the load. This helps to explain why the replacement of Viewpad nodes appears to have neutral to positive effects on E-SEAS performance.

The switching from mid-end devices to high-end and low-end devices in the MVR seems to cause an inverse effect for JEC and FWC schedulers. As Figure 5.4b and Figure 5.4c outline for JEC and FWC respectively, nodes are prioritized differently in order to exploit them. The highest amount of non finalized jobs, for both schedulers, is accumulated mostly in the device type with lowest MFLOPS (I5500), and a minor amount in mid-end devices. High-end devices finalized all their assigned jobs, meaning that, compared to the finalized jobs number that such devices present with E-SEAS, their computing capability is being underexploited. It explains why changing from mid-end devices to A100 devices negatively affects JEC and FWC performance.

These behaviors stems from the way rank values associated to each type of SMD vary according to each criterion formula. Analytically, with E-SEAS, for instance, for an I5500 node to rank higher than an A100 node, the former should have more than 8 times the remaining battery percentage of the A100. For example, by assuming that both nodes have not been assigned with any jobs yet, when the I5500 has approximately 40% of remaining battery, the A100 needs only 5% of its remaining battery to ranks equal. Alternatively, if both devices have the same remaining battery percentage, an I5500 would rank higher than an A100 if the latter has been assigned with 8 times the number of assigned jobs of the I5500. That makes an E-SEAS-based scheduler prone to saturate with jobs those nodes with high FLOPS first, then nodes with less FLOPS, and so on.

The opposite situation to that of E-SEAS occurs for node rankings derived from the application of JEC. Assuming the case in which the weakest and the strongest nodes are compared and none have been assigned any job yet, for an I5500 node to rank better than an A100 node, the former needs approximately more than 2.3 times the remaining battery percentage of the latter. Such condition is easier to meet for an I5500 node than when it is ranked with the E-SEAS. Thus, a JEC-based scheduler tend to assign jobs to the weak nodes first and then to the strong ones. However, this allocation criteria seems to create bigger underused periods of CPU cycles –i.e. sub-exploited states– than E-SEAS as ICC disparity among of nodes increases.

The behavior of FWC is quite similar to JEC. The major difference is that the multiplier factor of remaining battery percentage, which weak nodes need to overcome to rank better than strongest nodes, varies as the time passes. For this reason FWC suffers from a cold start effect. In numbers, considering short jobs, the multiplier starts with 1, because there is no information about the time a job lasts until the strongest nodes finalize their first allocated job. After that occurs, the multiplier factor changes to 2 because weakest nodes are still executing the first assigned job and the FWC ranking formula considers them to be in the middle of the execution time. In summary, the multiplier factor increases in multiple of 2 until 8 because the weakest node is 8 times weaker than the strongest one. It happens because the ranking formula of FWC is built upon pure dynamic components while E-SEAS and JEC have static components, i.e. MFLOPS and $JobEnergyConsumptionRate$ respectively in their ranking formulas.

5.2.2 Second phase: Evaluation of job stealing with battery-aware criteria

The experiments of this section aim at testing hypothesis 3.2 and hypothesis 3.2. The former states that due to system dynamics, inaccurate estimations of SMD computing potential and unknown tasks requirements partially break previous energy-aware task scheduling decisions creating sub-exploited slots of computing cycles that can be exploited through dynamic task re-balancing. Hypothesis 3.2, in turn, states that the improvement achieved by re-balancing is conditioned by the energy-aware scheduling decisions made in the past, namely the first scheduling phase.

This section presents the results of the second allocation phase, which are compared with those obtained in the first scheduling phase. A separate analysis is presented for each battery-aware criterion. The Round Robin scheduler is excluded from this set of experiments since it does not represent energy-aware scheduling decisions and it did not show competitive performance in the first allocation phase. Other change that attains to experiments is the elimination of job arrival rates from the variables of interest, decision whose justification was exposed at the end of Section 5.1.

Before explaining the results, some details of the notation used in the histograms is provided. First phase results are labeled with the name of the criterion –e.g. E-SEAS, JEC, FWC–. Second phase results notation are indicated with JS (referred to Job Stealing) followed by a letter that indicates the victim selection strategy, i.e., "W" for Worst Ranking-Aware Strategy and "B" for Best Ranking-Aware Strategy. Following the letter indicating the victim selection is another letter to indicate the offloading policy, i.e., "E" for Exponential policy and "F" for Fixed policy. When referring to the second phase, Job Stealing configuration or re-balancing phase results should be read the same.

Job Set	MVR	E-SEAS first phase	Second phase		
			JS_BF_E-SEAS	JS_WE_E-SEAS	JS_WF_E-SEAS
Short	MVR1	[10,042 - 10,147]	[10,042 - 10,147]	[10,042 - 10,147]	[10,042 - 10,147]
	MVR2	[9,790 - 10,047]	[9,774 - 10,020]	[9,790 - 10,047]	[9,790 - 10,047]
	MVR3	[10,162 - 10,472]	[10,151 - 10,472]	[10,162 - 10,472]	[10,162 - 10,472]
Long	MVR1	[1,624 - 1,647]	[1,626 - 1,643]	[1,624 - 1,647]	[1,623 - 1,647]
	MVR2	[1,590 - 1,629]	[1,587 - 1,607]	[1,590 - 1,629]	[1,590 - 1,629]
	MVR3	[1,642 - 1,699]	[1,643 - 1,699]	[1,655 - 1,715]	[1,655 - 1,716]

Table 5.4: Finalized jobs by E-SEAS based schedulers in the first and second phase

To ease the analysis and comparison of results, the visualization of each two-phase battery-aware scheduler performance for all scenarios is complemented with histograms. Figure 5.5a shows the performance of the first and second phase scheduling using E-SEAS for short job scenarios. Notice that re-balancing through any of the Job Stealing configurations using E-SEAS as criterion for victim selection does not improve the performance achieved during the first phase. The cause is explained after the performance description of JEC and FWC, but in principle it relates to in which node type the E-SEAS relies the completion of the highest amount of jobs during the first phase. The results associated to long job scenarios,

Job Set	MVR	JEC first phase	Second phase		
			JS_BF_JEC	JS_WE_JEC	JS_WF_JEC
Short	MVR1	[10,130 - 10,358]	[10,506 - 10,554]	[10,644 - 10,919]	[10,688 - 10,939]
	MVR2	[9,851 - 9,999]	[9,991 - 10,060]	[10,793 - 11,091]	[10,761 - 10,903]
	MVR3	[9,472 - 9,602]	[10,448 - 10,880]	[10,768 - 11,049]	[10,818 - 11,089]
Long	MVR1	[1,628 - 1,659]	[1,703 - 1,717]	[1,706 - 1,748]	[1,706 - 1,746]
	MVR2	[1,577 - 1,603]	[1,608 - 1,620]	[1,727 - 1,783]	[1,733 - 1,782]
	MVR3	[1,520 - 1,545]	[1,600 - 1,549]	[1,751 - 1,792]	[1,756 - 1,796]

Table 5.5: Finalized jobs by JEC based schedulers in the first and second phase

Job Set	MVR	FWC first phase	Second phase		
			JS_BF_FWC	JS_WE_FWC	JS_WF_FWC
Short	MVR1	[9,432 - 9,580]	[9,592 - 9,636]	[10,597 - 10,806]	[10,639 - 10,771]
	MVR2	[8,522 - 8,647]	[9,504 - 9,853]	[10,693 - 10,940]	[10,539 - 10,679]
	MVR3	[7,519 - 7,683]	[10,443 - 10,654]	[10,662 - 10,769]	[10,674 - 10,963]
Long	MVR1	[1,499 - 1,528]	[1,511 - 1,560]	[1,689 - 1,731]	[1,701 - 1,726]
	MVR2	[1,355 - 1,374]	[1,509 - 1,596]	[1,712 - 1,764]	[1,697 - 1,725]
	MVR3	[1,195 - 1,224]	[1,464 - 1,469]	[1,766 - 1,785]	[1,713 - 1,768]

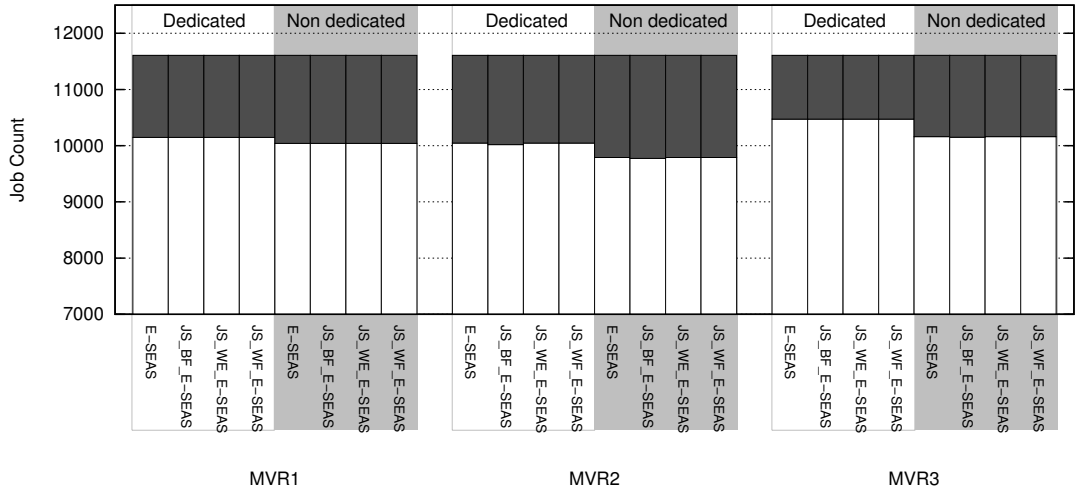
Table 5.6: Finalized jobs by FWC based schedulers in the first and second phase

depicted in Figure 5.5b, show that re-balancing slightly improved the performance achieved by the first phase, particularly for the MVR3 environment. In this case, JS_WE_E-SEAS and JS_WF_E-SEAS are the second phase configurations that slightly boost the performance of the first phase.

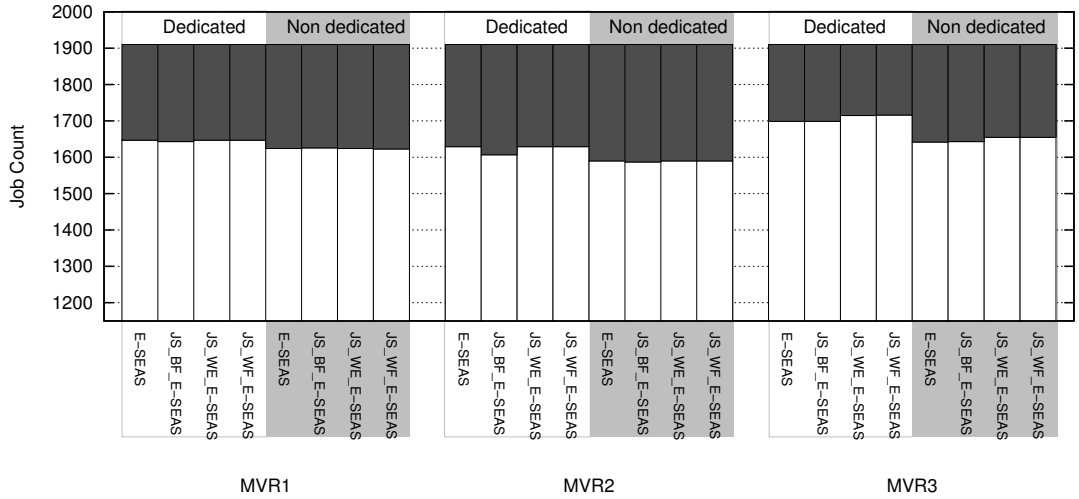
Figure 5.6 outlines the performance of the first and second phase scheduling using JEC. For all short jobs scenarios shown in Figure 5.6a, all JS configurations using JEC significantly improve the performance achieved during the first phase. The most efficient configurations are JS_WF_JEC and JS_WE_JEC. Furthermore, unlike JEC first phase results, the performance of these configurations does not decrease when including more powerful devices. It means that re-balancing using JEC suppresses that undesired effect, i.e., renders the scheduling performance less influenced by such type of devices, at least with the MVR instances studied. Regarding long job scenarios results, Figure 5.6b reveals that re-balancing has a similar effect to that observed in short job scenarios.

Figure 5.7 depicts the performance of the first and second scheduling phase using FWC. For short jobs scenarios, shown in Figure 5.7, all Job Stealing configurations improve the first phase using FWC. JS_WF_FWC and JS_WE_FWC obtained the highest performance boost while JS_BF_FWC the lowest one. Once again, long jobs scenarios results of Figure 5.7b follow the performance improvement pattern seen in short job scenarios.

The negligible performance boost of E-SEAS relates to the most overloaded nodes type and the time



(a) Short job scenarios



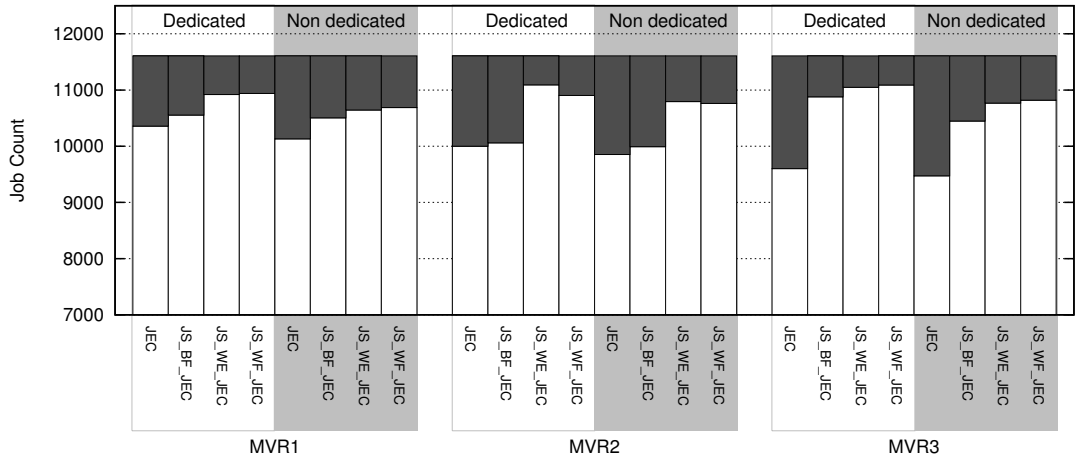
(b) Long job scenarios

Figure 5.5: E-SEAS performance: First and second phases

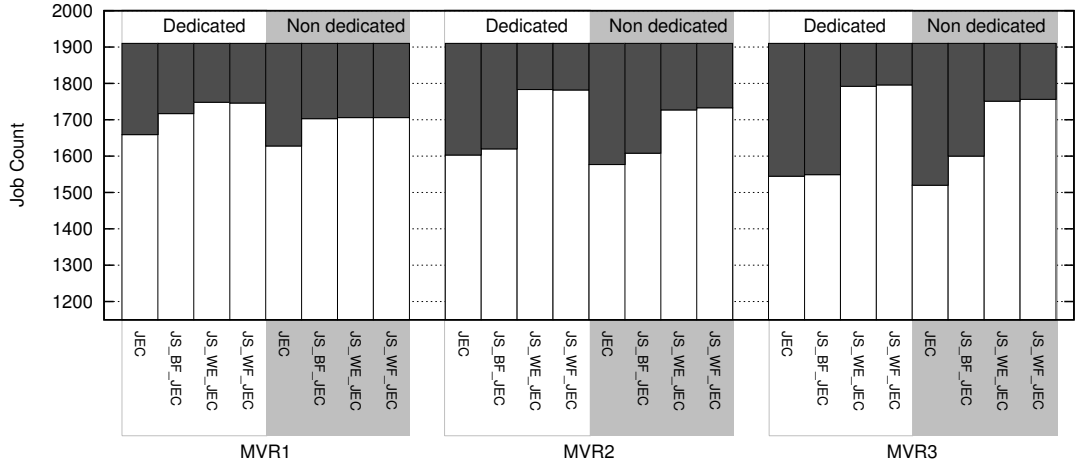
those nodes stay connected to the proxy. If overloaded nodes leave the MVR before least loaded or balanced nodes, then steals never occur or its frequency is low. Node departures, in the context of this thesis, is given by battery depletion events. Such effect is observed in the first phase scheduling using E-SEAS, where A100 nodes are overloaded. It is worth mentioning that it is assumed that queued (not started) jobs in nodes which leave the MVR are not re-scheduled and are considered non-finalized jobs. I leave that function to a fault tolerance mechanism that is out of the scope of this thesis.

5.2.2.1 Second phase concluding remarks

The performance boost all two-phase battery-aware schedulers introduce at the re-balancing phase shows that sub-exploited slots of computing cycles remain after the first phase job scheduling performed with an energy-aware scheduler, which is the statement of hypothesis 3.2. The statistical significance of

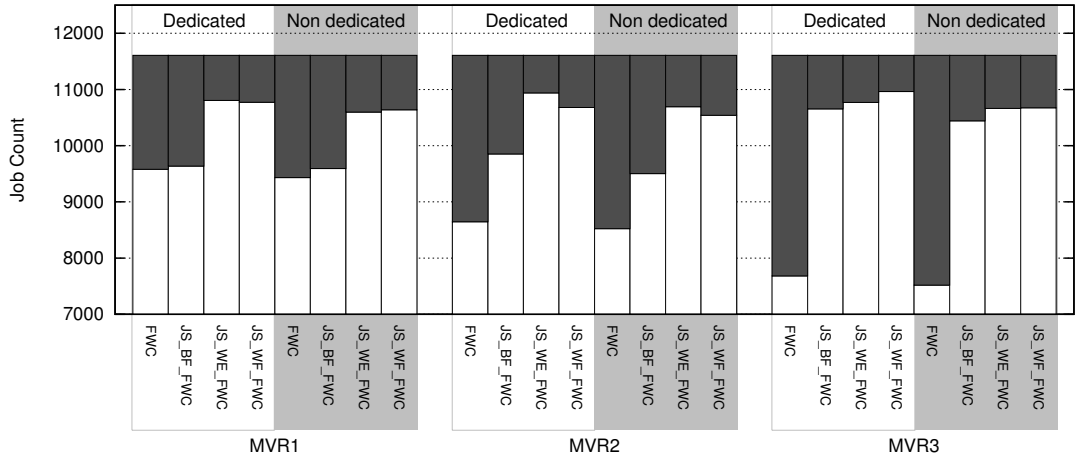


(a) Short job scenarios

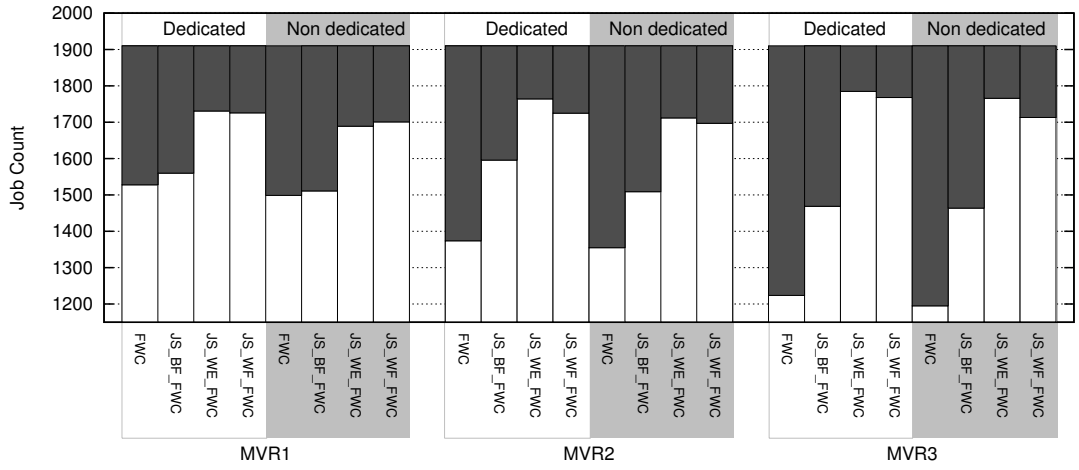


(b) Long job scenarios

Figure 5.6: JEC performance: First and second phases



(a) Short job scenarios



(b) Long job scenarios

Figure 5.7: FWC performance: First and second phases

such results is tested with a new set of Wilcoxon tests. In this case, paired samples derived from the performance of the battery-aware scheduler in the first phase and the performance achieved with a job stealing configuration for the same scenario. Since the second phase of each two-phase battery-aware scheduler was evaluated with three different job stealing configurations, independent tests are run for each configuration.

The null hypothesis (H_0) states that the mean number of finalized jobs achieved by a battery-aware scheduler during the first phase is equal to the mean number of finalized jobs achieved by a re-balancing phase that uses the same battery-aware criterion for victim selection. The alternative hypothesis (H_1) states that the mean number of finalized jobs achieved with a battery-aware scheduler differs from the mean number of finalized jobs achieved when the same criterion is used for victim selection in a re-balancing phase. The significance level or maximum acceptance of error type I is $\alpha = 0.01$.

Table 5.7 depicts the p-values of the tests for each two-phase battery-aware scheduler. Since p-values for all tests corresponding to JEC and FWC schedulers are less than the significance level selected then H_0

First phase	Re-balancing phase	P-value
E_SEAS	JS_WF_E-SEAS	0.0229090993544
	JS_WE_E-SEAS	0.0022177214642
	JS_BF_E-SEAS	0.0280561241492
JEC	JS_WF_JEC	0.0022177214642
	JS_WE_JEC	0.0022177214642
	JS_BF_JEC	0.0022177214642
FWC	JS_WF_FWC	0.0022177214642
	JS_WE_FWC	0.0022177214642
	JS_BF_FWC	0.0022177214642

Table 5.7: P-values of statistical tests for the first phase vs. the re-balancing phase

is rejected for these schedulers. It means that the first phase of these schedulers can be boosted with a re-balancing phase. The same does not hold for E-SEAS scheduler since p-values of the corresponding tests are over the significance level selected. There is evidence to accept hypothesis 3.2 although the analysis is not conclusive for the E-SEAS criterion.

In light of the performance boost registered with different re-balancing mechanisms, now is the turn to validate the sub-hypothesis 3.2 which, states that the improvement achieved by such re-balancing mechanism is conditioned by the energy-aware scheduling decisions made in the past, namely the first scheduling phase. The paired samples that constitute inputs of the new set of Wilcoxon tests are the number of finalized jobs achieved with a particular job stealing configuration whose first phase was performed by different battery-aware schedulers. Separate tests are run for re-balancing phase represented by JS_WF, JS_WE and JS_BF combined with all battery-aware criteria.

The null hypothesis H_0 is formulated as follows: the mean improvement, should be read as the final performance, achieved with a re-balancing mechanism does not vary with the energy-aware task scheduling decisions made in the first phase. The significance level or maximum acceptance of error type I is selected to be $\alpha = 0.01$.

The p-values resulting from the aforementioned tests are outlined in Table 5.8. With p-values under the threshold selected, H_0 is rejected for JS_WF and JS_WE. It means that the performance achieved by these re-balancing mechanisms is conditioned by the battery-aware scheduling decisions made in the first phase. The same does not hold for JS_BF re-balancing mechanism since p-values of the corresponding tests are over the threshold selected. There is evidence to accept the hypothesis 3.2 but the analysis is not conclusive for those job stealing configurations exploiting the JS_BF policy.

5.2.2.2 Second-phase battery-aware scheduling observations

As with first phase results, in this section it is presented a set of observations derived from the analysis of the second phase results reported in Section 5.2.2. The observations might be taken into account, in the

Re-balancing mechanism 1	Re-balancing mechanism 2	P-value
JS_WF_JEC	JS_WF_E-SEAS	0.00221772146424
	JS_WF_FWC	0.00221772146424
JS_WF_E-SEAS	JS_WF_FWC	0.00221772146424
JS_WE_JEC	JS_WE_E-SEAS	0.00221772146424
	JS_WE_FWC	0.00416350414614
JS_WE_E-SEAS	JS_WE_FWC	0.00221772146424
JS_BF_JEC	JS_BF_E-SEAS	0.02805612414920
	JS_BF_FWC	0.00221772146424
JS_BF_E-SEAS	JS_BF_FWC	0.08437944259400

Table 5.8: P-values of statistical tests for different rebalancing mechanisms

future, for the elaboration of further work hypotheses. From a cross two-phase battery-aware schedulers analysis the observations presented below arise.

Despite E-SEAS is among the criteria with the best performance during the first phase scheduling, such performance does not improve with a re-balancing mechanism, so that it does not keep the improvement pace achieved by the other two-phase battery-aware schedulers. Concretely, in short job scenarios, none of the two-phase E-SEAS-based schedulers, i.e., JS_WE_SEAS, JS_WF_E-SEAS or JS_BF_E-SEAS, performed better than all the other two-phase schedulers. Quantitatively, from 11,608 short jobs, the best two-phase E-SEAS-based scheduler finalizes around [5.32 - 8.99]% and [5.57 - 8.64]% less jobs than the best two-phase scheduler in dedicated and non-dedicated scenarios respectively. Seamlessly, from 1,910 long jobs, the best two-phase E-SEAS-based scheduler finalized around [4.19 - 8.06]% and [4.19 - 7.49]% less jobs than the best two-phase scheduler in dedicated and non-dedicated scenarios respectively.

Concerning the behavior of FWC-based schedulers, recall from the first phase results that the performance is quite under that of achieved by all other battery-aware competitors, and also suffers from the cold start effect. However, when outfitted with a re-balancing mechanism, the resulting two-phase FWC-based schedulers improve the performance, targeting higher number of finalized jobs than JS_WE_SEAS and JS_WF_E-SEAS in all scenarios. More specifically, JS_WE_FWC outperforms JS_WE_E-SEAS by [2.56 - 7.78]% and JS_WF_FWC to JS_WF_E-SEAS by [2.72 - 6.45]%.

In respect to the behavior of JEC-based schedulers, during the first phase it is within the criteria with the highest number of finalized jobs, but unlike E-SEAS, the re-balancing phase improves its performance. In fact, two-phase JEC-based schedulers, concretely JS_WE_JEC and JS_WF_JEC, are those that target the highest outcomes in all scenarios.

The followings are observations focused on highlighting energy implications of the best two-phase JEC--based schedulers. The usage of job stealing as re-balancing mechanism is likely to introduce administrative costs due to the network activity overhead produced by steal requests messages, which eventually

derive in job movements between stealer and victim nodes. The consequence of such administrative cost is that SMDs incur in energy waste -caused by extra networking activity- and underexploited CPU cycles -caused by the time a stealer waits until the stolen job is ready to be executed-. As a guide to differentiate between the two-phase schedulers with highest performance their energy implications are analyzed. The less steals a job stealing configuration produces per extra finalized job, the more efficient it is considered. For measuring steal efficiency it is used the *steal revenue* indicator. Such indicator, sixth column of Table 5.9, is defined as the extra finalized jobs over steals produced: $\frac{(extraFinalizedJobs)}{\#ofSteals}$. The $extraFinalizedJobs$ is the difference between the finalized jobs in the second phase minus the finalized jobs in the first phase. Table 5.9 (last column) shows the steal revenue along with the data used in the

Jobs size	MVR instance	SMD comp. capability	JEC 1st phase performance	2nd phase performance		# steals		Steals revenue	
				JS_WE	JS_WF	JS_WE	JS_WF	JS_WE	JS_WF
Short jobs	MVR1	Dedicated	10,358	10,919	**10,939**	4,674	619	0.120	**0.938**
		Non-dedicated	10,130	10,644	**10,688**	3,810	596	0.135	**0.981**
	MVR2	Dedicated	9,999	**11,091**	10,903	9,824	1,220	0.111	**0.741**
		Non-dedicated	9,851	**10,793**	10,761	6,189	1,015	0.152	**0.896**
	MVR3	Dedicated	9,602	11,049	**11,089**	16,138	1,914	0.090	**0.777**
		Non-dedicated	9,472	10,768	**10,818**	10,735	1,548	0.121	**0.870**
Long jobs	MVR1	Dedicated	1,659	**1,748**	1,746	221	104	0.402	**0.836**
		Non-dedicated	1,628	1,706	1,706	211	92	0.369	**0.848**
	MVR2	Dedicated	1,603	**1,783**	1,782	511	210	0.352	**0.852**
		Non-dedicated	1,577	1,727	**1,733**	378	171	0.397	**0.912**
	MVR3	Dedicated	1,545	1,792	**1,796**	687	319	0.360	**0.787**
		Non-dedicated	1,520	1,751	**1,756**	590	271	0.392	**0.871**

Table 5.9: Steal revenues comparison of best two-phase JEC-based schedulers

calculation of such value. The values of the fourth column and fifth columns (with its sub-columns) are number of finalized jobs. Fifth sub-columns show only values for JS_WE_JEC and JS_WF_JEC because these are the schedulers which, in average, achieve the highest number of finalized jobs.

The higher the steal revenue is, the more efficient the Job Stealing configuration is. JS_WF_JEC configuration outperforms JS_WE_JEC in all scenarios. This is because the stealing policy of JS_WF produces much less steals than that of JS_WE. The quantity of steals produced could serve to guide the selection of the best balanced scheduler for the case of communication channels with lower/higher latencies and energy costs.

Table 5.10 summarizes the improvement achieved by JS_WF_JEC, the two-phase scheduler with the highest performance. The data is analyzed from the perspective of the finite energetic constraints imposed by the used MVR instances. The FJ/E, expressed in e^{-4} notation, increases as total Joules of MVR instances decreases (see Table 5.1 for Joules information). It means that the two-phase scheduler for MVR1 achieves less FJ/E than for MVR2, and even less for MVR3. These relation is aligned with the

MVR instance		Dedicated				Non-Dedicated			
		Short		Long		Short		Long	
		1st Phase	2nd Phase	1st Phase	2nd Phase	1st Phase	2nd Phase	1st Phase	2nd Phase
MVR1 Joules: 6,445,008	FJ/E	16.07135	16.97283	2.57409	2.70907	15.71759	16.58338	2.52599	2.64701
	Improvement	**5.31%**		**4.98 %**		**5.22%**		**4.57%**	
MVR2 Joules: 5,035,752	FJ/E	19.9856	21.65119	3.1832	35.387	19.56212	21.3692	3.1316	3.44139
	Improvement	**8.29%**		**10.04%**		**8.46%**		**9.00%**	
MVR3 Joules: 3,626,496	FJ/E	26.4773	30.57773	4.2603	4.95244	26.1189	29.83045	4.1914	4.84214
	Improvement	**13.41%**		**13.97%**		**12.44%**		**13.44%**	

Table 5.10: FJ/E and energy exploitation improvement in % of 2nd phase with regard to 1st phase of the best average scheduler

fact that the $ACC/TotalEnergy$ of MVR1 is bigger than that of MVR2, and the latter bigger than that of MVR3. This efficiency in energy utilization is, on one side, naturally given by the energy properties of the devices composing the MVR, and on the other side, due to the two-phase scheduling criteria applied. The last affirmation stems from the fact that finishing less number of finalized jobs with the same energy means less energy efficiency, which is behavior shown by the other two-phase schedulers.

Finally, it is observed that JS_WF_JEC improvements of the second phase over the first phase increase as the target MVR is composed by more Acer A100 devices. Despite the JEC scheduler performance in the first phase decreases, the re-balancing phase not only compensates such decrements but boosts the performance to new maximums. This suggests that the two-phase scheduler as a whole adapts to MVRs with similar computing capacity but different $ACC/TotalEnergy$ without compromising the final performance.

5.3 Conclusions

Considering battery-related parameters for node ranking contributes to finalize more jobs than not considering them. Besides, from the observations made of battery-aware schedulers behavior in the first phase, it follows that E-SEAS and JEC schedulers are those that achieved the highest performance since they do not suffer from the cold start effect as FWC does. In all the scenarios exercised, E-SEAS outperforms Round Robin by at least 4.9% and JEC by at least 6.6%. The analysis of jobs that each type of SMD leaves unfinished reveals substantial differences on how E-SEAS and JEC achieve these gains. By using E-SEAS, the majority of unfinished jobs remain in nodes with the highest FLOPS. However, when JEC is used, the nodes with the greatest number of unfinished jobs are the ones with the lowest FLOPS values.

Moreover, furnishing battery-aware scheduling decisions of the first phase with re-balancing mechanisms improved the performance achieved of all schedulers. However, the magnitude of such improvement achieved with a job stealing technique is constrained by the battery-aware scheduling decisions of the first phase. The observations made on the second phase results reveal that the improvement of two-phase E-SEAS-based scheduler is marginal even with all re-balancing mechanisms since overloaded nodes, i.e., potential victims leave the MVR before stealers become idle so as to request for more jobs to complete. This is not the behavior of JEC-based and FWC-based schedulers whose performance in the first phase was considerably improved in the re-balancing phase. Nonetheless, despite FWC-based schedulers were very competitive in the second phase, their performance in the first phase was not. This outcome is particularly undesirable if the proxy has a downtime and the second phase cannot be initiated. For this reason, JEC-based schedulers might be considered the best average schedulers because they are competitive in the first phase and arise as the best in the second phase.

Aside from the performance achieved by JEC, from a qualitative perspective, its ranking formula includes the $JobEnergyConsumptionRate$ whose computation is flexible enough to consider the known aging problem of Li-ion batteries Takeno et al. (2005); Choi and Lim (2002). Unlike FLOPS used in the E-SEAS formula, the $JobEnergyConsumptionRate$ score associated to an SMD varies in time due to

battery decreasing lifetime. However, it would be interesting to investigate a way of computing of the $JobEnergyConsumptionRate$ score without requiring to profile a complete SMD discharge battery cycle.

From the second phase evaluation, it is observed that the WRAS strategy for selecting a victim obtained always better performance than the BRAS strategy. In respect to the job stealing policy, the Fixed policy shows values of steal revenue very superior to those achieved by the Exponential policy. The latter generates much more useless steals that should be avoided, e.g. as the size of job input increases, because they incur in higher energy misuse.

A general conclusion of this two-phase scheduling approach is that a re-balancing phase performed with the Job Stealing technique helps to increase the FJ/E achieved with a single phase scheduling, even when considering the energy footprint due to network activity of nodes. However, the node ranking criteria used for the first phase allocation influenced the performance boost that could be obtained or not with the re-balancing phase. More explicitly, the most effective instantiation of the re-balancing phase occurred when the first allocation phase left an active distributed pool of jobs. By active pool of jobs I mean jobs queued in nodes whose time to leave from the MVR is as far in time as possible. This increases the possibility to produce steals allowing the exploitation of underused periods of computing cycles to finalize more jobs.

Chapter 6

Conclusions

6.1 Concluding remarks

In this thesis, motivated by the continuous improvement of SMDs capabilities and the worldwide popularity of such devices, I presented an approach for scavenging computing capabilities of a group of SMDs. Related state-of-the-art RA mechanisms concentrate on addressing special features that characterize the problem of computing resource provision with SMDs. These special features, represented by the so-called singularities –user mobility, resource exhaustion and lack of ownership– pose new challenges to existing resource allocation mechanisms designed for fixed hardware. They introduce uncertainty, heterogeneity and fluctuating resource availability.

Many efforts to date have been focused on challenges associated to the user mobility singularity, e.g., for allocating tasks with hard execution time constraints, or tasks with data dependencies that heavily rely on SMD data transferring capabilities, or in the context of delay tolerant ad-hoc networks where tasks dissemination heavily depends on SMDs future contact and relative position. The negative effect of the user mobility singularity on tasks completion, however, is likely to be less harmful for the case of batch CPU-bound tasks, executed with the potential computing power of a stable number of SMDs coordinated through infrastructure-based networking support, which is the resource exploitation opportunity this thesis focuses on.

The scavenging of computing capabilities is always affected by the resource exhaustion singularity because the amount of work that can be completed is conditioned by the finite energy of SMDs batteries. The resource exhaustion has been addressed by many RA works but through optimal or near-optimal approaches, where tasks energy consumption on every candidate SMD is assumed to be provided as input or derived from SMDs and tasks characteristics. The last information, i.e., the time and/or computing requirements of tasks is not always available and it is hard, if not impossible, to estimate. In addition, the synthetic schemes that most of these works use to simulate the resource exhaustion singularity do not faithfully reflect the energy consumption of real SMDs.

The least studied singularity that conditions the exploitation of computing capabilities of SMDs is lack of ownership. This singularity reflects the shareable condition of computing resources by Grid tasks and

SMD owner processes. The only work Ghosh and Das (2010) that address the singularity suffers from the same applicability issues present in works which address resource exhaustion via optimal or sub-optimal approaches Shi et al. (2016); Ghasemi-Falavarjani et al. (2015); Li et al. (2015); Chunlin and Layuan (2014); Birje et al. (2014); Mtibaa et al. (2013); Wei et al. (2013); Shi et al. (2012); Furthmüller and Waldhorst (2012); Vaithiya and Bhanu (2012); Comito et al. (2011); Jang and Lee (2011); Ilavarasan and Manoharan (2010); Park et al. (2003), and the evaluation do not contemplate neither task execution degradation nor energy depletion due to SMDs owner interaction.

This thesis consists in a practical online two-phase scheduling approach for exploiting computing capabilities of SMDs coordinated via infrastructure-based networking support. The first-phase distributes arriving tasks to SMDs, where the computing capability of the latter is estimated using *criteria* that consider energy-related parameters. Moreover, to avoid sub-exploited slots of computing cycles caused by tasks heterogeneity, fluctuation of resource availability -due to owner interaction-, lack of up-to-date information and/or battery level estimation errors, a re-balancing scheme is provided in a second phase. This second or re-balancing phase is materialized with Job Stealing techniques, whose effectiveness have been studied in many past works in the area and is also included in several middlewares for distributed computing. The energy-aware criteria also plays a vital role in making these job stealing techniques effective.

The criteria combine easy-to-obtain energy and computing related features of SMDs. E-SEAS, for example, uses remaining battery charge data, available via the Android BatteryManager API, and FLOPS. The latter is used to score computing capability of nodes in HPC systems and is measurable in SMDs using benchmarks for Android (e.g. Linpack for Android in the Play Store). JEC is another energy-aware criteria proposed for the first phase that, instead of FLOPS information, uses a custom score computed based on benchmarking test results and battery capacity declared by the SMD manufacturer. The third energy-aware criterion, FWC, does not rely on pre-computed values to rate the computing capability of an SMD but infers it through execution time of finalized tasks.

One work hypothesis was whether ranking SMDs using criteria built upon energy-related parameters helps to better estimate the computing potentials of a group of SMDs than using a non energy-aware criterion. To validate the hypothesis, the number of finalized jobs was utilized as the metric for evaluating scheduling performance. The round robin scheduler was selected as a representative of existing non-energy aware schedulers. Schedulers performance was compared based on identical simulated conditions, i.e., scenarios. The scenarios design was driven by heterogeneity and include SMDs with different energy/computing features which, in turn, vary during runtime due to owner interaction effects. In addition to the heterogeneity supported on SMDs internal features, the scenarios include heterogeneity at the MVR level -via combination of different SMDs-, different tasks sizes and varied tasks arrival rates. Tasks scheduling with the energy-aware criteria proposed for the first scheduling phase outperformed to the non-energy aware scheduler in number of finalized jobs.

A second work hypothesis was whether the re-balancing phase boosts the performance achieved in the first scheduling phase. The results reveal that the number of finalized tasks scheduled in a first phase with JEC or FWC can be incremented with a re-balancing phase. However, such increment is not observed when the first scheduling phase is performed with E-SEAS. When investigating the E-SEAS behavior,

it was observed that overloading high throughput SMDs in the first phase causes these nodes to deplete their energy before mid and low throughput SMDs. The consequences of such behavior is detriment to job stealing effectiveness and hence the load re-balancing phase.

The SCI/SCI-E journal publications directly related to this research produced so far are listed below. One article not listed here has passed the second round of review in a SCI-indexed journal –Journal of Simulation (Springer)–, and is under review.

- Hirsch, M., Rodriguez, J. M., Zunino, A., & Mateos, C. (2016). Battery-aware centralized schedulers for CPU-bound jobs in mobile Grids. Pervasive and Mobile Computing, 29, 73-94. **Indexed SCI-E**.
- Hirsch, M., Rodriguez, J. M., Mateos, C., & Zunino, A. (2017). A two-phase energy-aware scheduling approach for CPU-intensive jobs in mobile Grids. Journal of Grid Computing, 15 (1), 55-80. **Indexed SCI-E**.
- Hirsch, M., Rodriguez, A., Rodriguez, J. M., Mateos, C., & Zunino, A. (2017). Spotting and Removing WSDL Anti-pattern Root Causes in Code-first Web Services Using NLP Techniques: A Thorough Validation of Impact on Service Discoverability. Computer Standards & Interfaces, 56, 116-133. **Indexed SCI-E**.

Likewise, two articles published in conferences are listed below:

- Hirsch, M., Mateos, C., Rodriguez, J. M., Zunino, A., Garí, Y. & Monge, D. A performance comparison of data-aware heuristics for scheduling jobs in mobile Grids. XLIII CLEI (2017) - SLIHS (Simposio Latinoamericano de Infraestructura, Hardware y Software). Córdoba, 2017.
- Hirsch, M., Rodriguez, A., Rodriguez, J. M., Mateos, C., Zunino, A., & Ordiales Coscia, J. A tool for building retrievable code-first Web Services. 2014 IEEE Biennial Congress of Argentina (ARGENCON). San Carlos de Bariloche, 2014. IEEE Computer Society.

6.2 Limitations

During the development of this thesis, some limitations worth to be discussed have been identified.

First, the two-phase scheduling approach has been evaluated under conditions of resources wide-view knowledge and centralized single-hop communication with SMDs, which are features of infrastructure-based networks. The behavior of the approach under the operation conditions of ad-hoc networks need to be investigated. Cluster heads node selection Yang et al. (2010) can be applied to assign certain SMDs with the proxy role. Then, these nodes would act as resource information sink, i.e., nodes to which resource providers SMDs in the vicinity route resource update messages. In this way, resource providers can also play the role of resource requesters and communicate with the cluster head to know the most convenient SMD to ask for task execution service. In this scheme, with knowledge about available

resources in the vicinity, a cluster head would apply the first-phase of the proposed scheduling approach. The application of the second phase could be performed by the cluster head or delegated to nodes in the vicinity.

Second, the energy-aware criteria proposed to instantiate the first phase of the scheduling approach, which were also used for victim selection in the second phase, were designed with CPU cycles requirements of jobs in mind only. Despite that jobs were associated to input and output data requirements, data sizes are assumed to be fixed for all jobs and SMDs energy consumed per transferring data unit is also fixed for all SMDs. To add more realism concerning this aspect, data-aware heuristics have started to be investigated for scheduling jobs with hybrid requirements Hirsch et al. (2017a). The performance of several heuristics inspired in traditional ones, including max-min, min-min, Minimum Completion Time (MCT), were studied to schedule jobs of varying input/output data and CPU cycles requirements. The heuristics operate without assuming jobs information related to CPU requirements. Instead, they just assume to know job data transfer information and SMDs connection quality parameters to schedule jobs.

The performance of the two-phase approach has been evaluated under simulated conditions. Although simulation is the most adopted evaluation methodology in the area, and the in-vitro simulation scheme endows the evaluation with a distinctive quote of realism, an in-vivo performance evaluation to enforce the obtained results is nevertheless desirable. Nonetheless, efforts in my research group towards implementing the proxy-based architecture and the ranking-based scheduling criteria are underway Perez Campos et al. (2017). In the aforementioned work, an empirical assessment of the SEAS against round robin and random approaches was performed, and the results obtained confirm the simulated performance reported in the original SEAS paper Rodriguez et al. (2010).

Lastly, the two-phase approach lacks a mid-term scheduling component to avoid saturating an MVR with jobs that have low chances to finalize. In other words, the load represented by all jobs received by a proxy is re-distributed by the two-phase approach based on computing-energy related characteristics of SMDs offering computing resources. However, until an SMD does not loose connection, e.g., due to battery depletion, the jobs waiting for execution in the queue of that SMD will wait despite having no chance -neither locally at the SMD, nor globally at the MVR- to start executing. Outfitting the current short-term two-phase scheduling approach with a mid-term scheduling would be useful to automatically regulate the acceptance of new jobs into the MVR. With such a mechanism the number of non-finalized jobs and the turnaround time of jobs could be reduced since jobs which cannot be finalized in one MVR could be redirected to a nearby MVR in a kind of inter-MVR offloading process.

6.3 Future works

This work is expected to further develop in several directions. Owner behavior is a concern intrinsically related to SMDs resource exploitation whose study and implications are detailed in Section 6.3.1. Many RA works assume perfect task knowledge and up-to-date resource availability information as pre-conditions to run batch scheduling logic. Given SMDs heterogeneity and volatility of resource availability, such conditions are hard to obtain and subject to frequent changes. Then, a question that arises is

how valuable this information is for batch RA works to get advantage over online RA works providing that the latter compute a task mapping solution with less complexity than the former. This line of future work is detailed in Section 6.3.2. To accelerate the performance comparison of different RA works and hence rapidly advance research in the area, standard software platforms and tools are needed. This topic is discussed in Section 6.3.3. Issues about addressing incentives, security and privacy concerns are discussed in Section 6.3.4. Finally, the potential convergence of SMDs and mobile Grids and recent service-oriented computing paradigms is highlighted in Section 6.3.5.

6.3.1 Allocating resources being aware of SMD owner behavior

The map depicting the singularities which most RA works has been focused on, summarized in Table 2.3, shows that much of the research has been conducted to address challenges posed by, in first place, exhaustible resources of SMDs and, in the second place, user mobility. More investigation is needed with regard to the lack of ownership singularity. It involves not only the way it affects external task execution, but the way external tasks execution is expected to alter SMD owner experience.

The work developed in the context of this thesis can be extended to outfit the two-phase scheduling approach with lack of ownership-aware capabilities. It means not only to consider the singularity within experimental design, but to develop indicators, algorithms and techniques aligned with the goals summarized in Table 2.6: to target a higher exploitation of available resources and achieve a balanced exploitation of resources considering owner's future needs. Outfitting schedulers with knowledge derived from SMDs owner usage profile would therefore help in:

- *Increase the efficiency of resource scavenging*: the MVR productivity could benefit from indicators that rate SMDs potential computing capabilities considering not only hardware specifications but also owner behavior and future computing needs. Indicators might exploit, for example, context-aware information Pejovic and Musolesi (2015); Makris et al. (2013), human battery charge behaviors Rahmati et al. (2007), applications interaction patterns Wagner et al. (2014), etc.

- *Achieve seamless participation*: a system whose execution capability relies on computing resources of SMDs, depends on SMDs owner participation. In cooperative scenarios, e.g., translation at the museum Huerta-Canepa and Lee (2010), participation is given in a natural way, because owners typically share common objectives, and benefit from the same results. In non-cooperative scenarios, owners are encouraged to participate, i.e., offer their unused resources, in exchange of monetary credits or some sort of revenue. Without awareness of future owner activities, the system could tend to greedily use as much resources as available, causing an aggressive drain of SMDs battery and making them unusable for satisfying near future owner's personal needs. Irrespective of the revenue that the system would pay for the usage of resources, device owner willingness to participate as resource provider could be threaten by such kind of system behavior. The owner could perceive that the participation is not as seamless he/she would expect.

6.3.2 Studying the impact of accurate task information

The effectiveness of a high percentage of state-of-the-art RA proposals for MVR heavily depends on accurate input of tasks characteristics and requirements. In general, such proposals expect the task originator to submit tasks indicating *timing information* referring to either the execution time of the task in a particular SMD or the time before which the task result should be computed (task deadline). The time is associated to the computing capabilities, i.e., CPU power, of SMDs and depending on tasks requirements, it may also include information related to data transferring time. Additionally, RA mechanisms addressing the resource exhaustion singularity often rely on tasks energy consumption information which is usually assumed or derived based on the time a task spends using resources.

Predicting task times is a complex task itself, when not impossible, for the general case Wilhelm et al. (2008). Averaging historical execution times of a task can be, at first, a practical workaround for predicting future task behavior. However, such strategy is not robust enough to embrace new tasks codes or even legacy code considering that slight variations in data input can result in very different execution times. To make things worse, the heterogeneity and dynamic nature of SMDs resource provision are important sources of execution time variability. In presence of such volatile execution conditions, addressing the problem of obtaining accurate task information is crucial.

This thesis can be extended to contribute in the aforementioned direction by targeting a comparison with other RA approaches that share a minimum common set of assumptions. This encompasses, for instance, RA proposals which despite depending on accurate task information to operate, include some logic for detecting and reacting to changes in the resources availability Viswanathan et al. (2015), or approaches focused on different singularities combination but assuming similar jobs characteristics. Such cross comparison require finding common metrics for measuring performance, adapting current scheduler criteria for supporting other performance metrics –e.g., flowtime, makespan, or fairness in resource utilization–, and reproducing experimental variables of other RA mechanisms –e.g., disconnection–. Moreover, due to the disparity in the range of experimental variables considered and the quite often scarce provision of details about all parameters needed to reproduce experimental conditions, it can be also necessary to implement algorithms and/or indicators used by other RA mechanisms. In relation to the efforts of reproducing experimental conditions of other works and/or compare the performance of other RA mechanisms, it is evident the need for a common experimentation platform. This point is further developed below.

6.3.3 Unifying and enhancing the evaluation methodologies

The complex nature of SMDs clusters, given by their heterogeneity, dynamism and hard-to-reproduce scenarios, has pushed simulation as the most accepted practice for evaluating RA performance in the area. However, simulation software toolkits which are publicly available for use and modification, with features for investigating RA related concerns associated to all singularities, different type of tasks and communication supports, do not exist yet. Therefore, the community efforts for producing experimental results go into multiple ways. For instance, there is software derived from isolated programming efforts Birje et al. (2014) that target specific instances and parameters of the RA problem, instantiations of

general simulation frameworks -e.g., JAVASim Chunlin and Layuan (2014), MATLAB Singh and Raza (2017); Li et al. (2015), SIMGrid Lee et al. (2014); Vaithiya and Bhanu (2012)- and purpose-specific, well-known software developed for other related research areas that provide support to only a subset of features present in mobile Grid environments, e.g., NS-3, OMNet++. There are also several works published in prestigious forums that do not specify the software nor the techniques employed for simulating the performance of RA mechanisms. Moreover, SMDs representation of internal relations, e.g., between tasks execution and battery consumption, are quite often oversimplified through the use of synthetic functions and artificial (unrealistic) values. In addition, algorithms, techniques, models, and/or parameters employed for simulating the effects of SMDs singularities are shallowly explained and/or not referenced. An open source simulation platform that leverages up-to-date information derived from mobile device datasets -e.g., Device Analyzer Wagner et al. (2014), CRAWDAD Kotz et al. (2009), Human Activity Recognition using Smartphones Anguita et al. (2013)- for simulating a wide variety of resource exploitation opportunities is necessary. Such a platform would help to:

- *Promote cross-validation tests*: by utilizing an unified simulation platform, i.e., a shared set of features and input data for modeling devices behavior, the effectiveness of new RA mechanisms could be easily compared against others previously proposed. This would promote a healthy competition among researchers -as in other mature computing research fields like Information Retrieval or even traditional High-Performance Computing- looking for the most effective RA mechanism depending on the exploitation context. In addition, the fact that an RA mechanism be included in several studies would help to stress the evaluation of the algorithm and get insights on its behavior under different resource exploitation scenarios and situations.

- *Accelerate the production of results in the area*: the research community could accelerate the development of RA mechanisms by benchmarking their efforts via one (or few) common and massively used simulation platform(s). Topics such as assistance for setting up simulation environments, automated configurations, extensions to the framework for supporting new functionality, creation of software libraries and applications for filtering information from available data-sets and/or generating new data-sets, how to integrate modules from other simulators, amongst many others, could be subject of fruitful discussions in special forums.

6.3.4 Addressing incentive, security and privacy concerns

There are other important orthogonal concerns –alongside the three singularities discussed– that should be addressed to ensure that SMD cluster usage goes mainstream. Particularly, such concerns are how to systematically lure users into offering their SMDs resources to external applications, how to deal with potential security treats, and how to preserve privacy in the general sense.

From the reviewed works it follows that economic models for compensating SMDs resources usage and incentive owner participation is a topic that deserves more attention by the research community. The price that should be paid for the usage of resources is an open issue. Only few works, from those analyzed in this survey, address this concern Ghasemi-Falavarjani et al. (2015); Birje et al. (2014); Chunlin and

Layuan (2014). In Ghasemi-Falavarjani et al. (2015), the authors outfit the RA they propose with a virtual credit mechanism to reward SMDs owners who contribute with resources. The price of resource usage is calculated by measuring the execution time and energy consumed by a known device. A set of experiments were designed to show the impact in energy consumption of SMDs owners with selfish and altruist profiles. In Chatzopoulos et al. (2016) the authors present OPENRP, a reputation middleware for opportunistic crowd computing where a similar analysis of SMDs owners participation profiles is performed. In Birje et al. (2014); Chunlin and Layuan (2014) the price of resources is determined through a non-cooperative bargaining game model that resource providers and resource consumers play, intermediated by resource brokers. A limitation of those proposals is that resource contributors can know how much credits will receive in exchange of their resources only if detailed tasks requirements are known before tasks are assigned by the RA. As pointed out in Section 6.3.2, if an RA mechanism strongly depends on the availability of such information, its application scope is reduced. In absence of such information, there is no way to valuate a provider, i.e., new indicators need to be designed to rate SMDs. The rating might involve weighting some SMD features more than others and that must be in accordance with the system objectives and tasks characteristics. For instance, for a system whose objective is to reduce makespan, nodes that complete tasks quickly can be more valuable than those which complete tasks in an energy efficient way. By contrast, a distributed computing system which prioritizes the availability over the speed of execution services could rate higher SMDs with long battery life than those which provide high computing throughput.

In a similar line, security and privacy concerns are still not being addressed from the RA logic. Security and privacy involve mechanisms that assure transparency in the usage of resources and prevent misusing effects for both counterparts of the system, i.e., resource providers and resource consumers. In other words, resource consumers should be assured that tasks input (data to be processed, execution code) and results will not be altered neither by intruders in a network nor by the SMD owner who voluntarily offer the resources. From the resource providers point of view, the system would need to assure that tasks execution will not use or alter personal information and documents from the SMD owner and violate his/her privacy. Although techniques such as data encryption Pasupuleti et al. (2016) and operating system level virtualization Wessel et al. (2015) can help in achieving these objectives, the time and energy consumption of applying such solutions has not been studied in the context of RA mechanisms, e.g., as overhead costs.

6.3.5 The FaaS concept to leverage SMDs resources

Cloud Computing platforms deliver hardware and software resources to clients through services according to three classical provision models named Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). These models also determine the abstraction level at which resources are provided. Broadly, at the lowest layer, i.e., the IaaS layer, a combination of hosting, hardware provisioning and basic services needed to run a Cloud are provided. The PaaS layer, which sits on top of the IaaS layer, offers computing platform and software application deployment services to enterprises. On top of the PaaS layer is the SaaS layer through which vendors host applications and services to make them available to their customers.

With the popularity of Cloud services, the spectrum of potential services has become richer and numerous features started to be offered as services Varghese and Buyya (2017). An example of such features are *functions* offered as a service (FaaS) from the PaaS layer. Functions are the building blocks of the serverless computing model, which promotes event-based invocations of lightweight pieces of remote codes, and represents a simplified way for non-expert users to benefit from distributed computing since it eliminates the need to deal with the complexity of cluster management and configuration tools Jonas et al. (2017); Spillner et al. (2017).

Functions lightweight characteristic, i.e., low overhead configuration and execution runtime, and stateless codes make them perfect candidates for being deployed and executed with the limited resources of SMDs. Under this scheme of code execution, SMDs willing to offer resources such as CPU cycles or sensors could for example host associated functions and announce the function list they support. Such announcements, which could take place at the time an SMD joins an MVR, might involve describing through a service description artifact each function purpose, invocation parameters and return types. In this context, functions might inherit the same discoverability and understandability issues of traditional Web Services Hirsch et al. (2017b, 2014).

Bibliography

Abolfazli, S., Sanaei, Z., Ahmed, E., Gani, A., and Buyya, R. (2014). Cloud-based augmentation for mobile devices: motivation, taxonomies, and open challenges. *Communications Surveys & Tutorials, IEEE*, 16(1):337–368.

Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., and Cayirci, E. (2002). Wireless sensor networks: a survey. *Computer networks*, 38(4):393–422.

Akyildiz, I. F., Xie, J., and Mohanty, S. (2004). A survey of mobility management in next-generation all-ip-based wireless systems. *IEEE Wireless communications*, 11(4):16–28.

Aldunate, R., Larson, G., Nussbaum, M., Ochoa, S., and Herrera, O. (2006). Understanding the role of mobile ad hoc networks in non-traditional contexts. *Mobile and Wireless Communication Networks*, pages 199–215.

Alonso-Monsalve, S., García-Carballeira, F., and Calderón, A. (2017). Combos: A complete simulator of volunteer computing and desktop grids. *Simulation Modelling Practice and Theory*, 77:197–211.

Anguita, D., Ghio, A., Oneto, L., Parra, X., and Reyes-Ortiz, J. L. (2013). A public domain dataset for human activity recognition using smartphones. In *ESANN*.

Arslan, M. Y., Singh, I., Singh, S., Madhyastha, H. V., Sundaresan, K., and Krishnamurthy, S. V. (2012). Computing while charging: building a distributed computing infrastructure using smartphones. In *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, pages 193–204. ACM.

Ba, H., Heinzelman, W., Janssen, C.-A., and Shi, J. (2013). Mobile computing-a green computing resource. In *Wireless Communications and Networking Conference (WCNC), 2013 IEEE*, pages 4451–4456. IEEE.

Balasubramanian, N., Balasubramanian, A., and Venkataramani, A. (2009). Energy consumption in mobile phones: a measurement study and implications for network applications. In *9th ACM SIGCOMM conference on Internet measurement conference*, pages 280–293. ACM.

Balazinska, M. and Castro, P. (2003). Characterizing mobility and network usage in a corporate wireless local-area network. In *Proceedings of the 1st international conference on Mobile systems, applications and services*, pages 303–316. ACM.

Banavar, G., Beck, J., Gluzberg, E., Munson, J., Sussman, J., and Zukowski, D. (2000). Challenges: an application model for pervasive computing. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 266–274. ACM.

Banks, J., CARSON II, J. S., Barry, L., et al. (2005). *Discrete-event system simulation fourth edition*. Pearson.

Birje, M. N., Manvi, S. S., and Das, S. K. (2014). Reliable resources brokering scheme in wireless grids based on non-cooperative bargaining game. *Journal of Network and Computer Applications*, 39:266–279.

Black, M. and Edgar, W. (2009). Exploring mobile devices as grid resources: Using an x86 virtual machine to run boinc on an iphone. In *Grid Computing, 2009 10th IEEE/ACM International Conference on*, pages 9–16. IEEE.

Büsching, F., Schildt, S., and Wolf, L. (2012). Droidcluster: Towards smartphone cluster computing – the streets are paved with potential computer clusters. In *2012 32nd International Conference on Distributed Computing Systems Workshops*, pages 114–117.

Buyya, R. and Murshed, M. (2002). GridSim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. *Concurrency and Computation: Practice and Experience*, 14(13):1175–1220.

Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A. F., and Buyya, R. (2011). Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience*, 41(1):23–50.

Camp, T., Boleng, J., and Davies, V. (2002). A survey of mobility models for ad hoc network research. *Wireless communications and mobile computing*, 2(5):483–502.

Casanova, H. (2001). Simgrid: a toolkit for the simulation of application scheduling. In *Proceedings First IEEE/ACM International Symposium on Cluster Computing and the Grid*, pages 430–437.

Castro, M. C., Kassler, A. J., Chiasserini, C.-F., Casetti, C., and Korpeoglu, I. (2010). Peer-to-peer overlay in mobile ad-hoc networks. In *Handbook of Peer-to-Peer networking*, pages 1045–1080. Springer.

Chang, W.-Y. (2013). The state of charge estimating methods for battery: A review. *ISRN Applied Mathematics*, 2013.

Chatzopoulos, D., Ahmadi, M., Kosta, S., and Hui, P. (2016). Openrp: a reputation middleware for opportunistic crowd computing. *IEEE Communications Magazine*, 54(7):115–121.

Chengetanai, G. and O'Reilly, G. B. (2015). Survey on simulation tools for wireless mobile ad hoc networks. In *2015 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, pages 1–7. IEEE.

Choi, S. S. and Lim, H. S. (2002). Factors that affect cycle-life and possible degradation mechanisms of a li-ion cell based on licoo2. *Journal of Power Sources*, 111(1):130 – 136.

Chunlin, L. and Layuan, L. (2014). Exploiting composition of mobile devices for maximizing user qos under energy constraints in mobile grid. *Information Sciences*, 279:654 – 670.

Comito, C., Falcone, D., Talia, D., and Trunfio, P. (2011). Energy efficient task allocation over mobile networks. In *Dependable, Autonomic and Secure Computing (DASC), 2011 IEEE Ninth International Conference on*, pages 380–387. IEEE.

Conti, M., Giordano, S., May, M., and Passarella, A. (2010). From opportunistic networks to opportunistic computing. *IEEE Communications Magazine*, 48(9):126–139.

Ding, N., Wagner, D., Chen, X., Pathak, A., Hu, Y. C., and Rice, A. (2013). Characterizing and modeling the impact of wireless signal strength on smartphone battery drain. In *ACM SIGMETRICS Performance Evaluation Review*, volume 41, pages 29–40. ACM.

Duan, L., Kubo, T., Sugiyama, K., Huang, J., Hasegawa, T., and Walrand, J. (2014). Motivating smartphone collaboration in data acquisition and distributed computing. *IEEE Transactions on Mobile Computing*, 13(10):2320–2333.

Elmangoush, A., Corici, A., Al-Hezmi, A., and Magedanz, T. (2015). 11 - mobility management for machine-to-machine (m2m) communications. In Antón-Haro, C. and Dohler, M., editors, *Machine-to-machine (M2M) Communications*, pages 187 – 206. Woodhead Publishing, Oxford.

Falaki, H., Mahajan, R., Kandula, S., Lymberopoulos, D., Govindan, R., and Estrin, D. (2010). Diversity in smartphone usage. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, pages 179–194. ACM.

Fernando, N., Loke, S. W., and Rahayu, W. (2013). Mobile cloud computing: A survey. *Future Generation Computer Systems*, 29(1):84–106.

Fernando, N., Loke, S. W., and Rahayu, W. (2016). Computing with nearby mobile devices: a work sharing algorithm for mobile edge-clouds. *IEEE Transactions on Cloud Computing*. In press.

Ferretti, S., Ghini, V., and Panzieri, F. (2016). A survey on handover management in mobility architectures. *Computer Networks*, 94:390 – 413.

Furthmüller, J. and Waldhorst, O. P. (2010). A survey on grid computing on mobile consumer devices. *Handbook of Research on P2P and Grid Systems for Service-Oriented Computing, IGI-Global*, pages 313–363.

Furthmüller, J. and Waldhorst, O. P. (2012). Energy-aware resource sharing with mobile devices. *Computer Networks*, 56(7):1920–1934.

Ghasemi-Falavarjani, S., Nematbakhsh, M., and Ghahfarokhi, B. S. (2015). Context-aware multi-objective resource allocation in mobile cloud. *Computers & Electrical Engineering*, 44:218–240.

Ghosh, P. and Das, S. K. (2010). Mobility-aware cost-efficient job scheduling for single-class grid jobs in a generic mobile grid architecture. *Future Generation Computer Systems*, 26(8):1356–1367.

González-Castaño, F. J., Vales-Alonso, J., Livny, M., Costa-Montenegro, E., and Anido-Rifón, L. (2003). Condor grid computing from mobile handheld devices. *SIGMOBILE Mob. Comput. Commun. Rev.*, 7(1):117–126.

Gupta, H., Vahid Dastjerdi, A., Ghosh, S. K., and Buyya, R. (2017). ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments. *Software: Practice and Experience*, 47(9):1275–1296.

Habak, K., Ammar, M., Harras, K. A., and Zegura, E. (2015). Femto clouds: Leveraging mobile devices to provide cloud service at the edge. In *Cloud Computing (CLOUD), 2015 IEEE 8th International Conference on*, pages 9–16. IEEE.

Hirsch, M., Mateos, C., Rodriguez, J. M., Zunino, A., Garı, Y., and Monge, D. A. (2017a). A performance comparison of data-aware heuristics for scheduling jobs in mobile grids. In *XLIII CLEI - Inpress*, page in press. IEEE.

Hirsch, M., Rodriguez, A., Rodriguez, J. M., Mateos, C., and Zunino, A. (2017b). Spotting and removing wsdl anti-pattern root causes in code-first web services: A thorough evaluation of impact on service discoverability. *Computer Standards & Interfaces*.

Hirsch, M., Rodriguez, A., Rodriguez, J. M., Mateos, C., Zunino, A., and Coscia, J. L. O. (2014). A tool for building retrievable code-first web services. In *Biennial Congress of Argentina (ARGENCON), 2014 IEEE*, pages 165–170. IEEE.

Hirsch, M., Rodríguez, J. M., Mateos, C., and Alejandro, Z. (2017c). A two-phase energy-aware scheduling approach for cpu-intensive jobs in mobile grids. *Journal of Grid Computing*, 15(1):55–80.

Hirsch, M., Rodriguez, J. M., Zunino, A., and Mateos, C. (2016). Battery-aware centralized schedulers for cpu-bound jobs in mobile grids. *Pervasive and Mobile Computing*, 29:73–94.

Hoque, M. A., Siekkinen, M., Khan, K. N., Xiao, Y., and Tarkoma, S. (2015). Modeling, profiling, and debugging the energy consumption of mobile devices. *ACM Comput. Surv.*, 48(3):39:1–39:40.

Hu, Y. and Yurkovich, S. (2012). Battery cell state-of-charge estimation using linear parameter varying system techniques. *Journal of Power Sources*, 198:338–350.

Huerta-Canepa, G. and Lee, D. (2010). A virtual cloud computing provider for mobile devices. In *Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond*, page 6. ACM.

Hussain, H., Malik, S. U. R., Hameed, A., Khan, S. U., Bickler, G., Min-Allah, N., Qureshi, M. B., Zhang, L., Yongji, W., Ghani, N., et al. (2013). A survey on resource allocation in high performance distributed computing systems. *Parallel Computing*, 39(11):709–736.

Ilavarasan, E. and Manoharan, R. (2010). High performance and energy efficient task scheduling algorithm for heterogeneous mobile computing system. *International Journal of Computer Science & Information Technology*, 2(2):10–27.

Ilavarasan, E. and Thambidurai, P. (2007). Low complexity performance effective task scheduling algorithm for heterogeneous computing environments. *Journal of Computer sciences*, 3(2):94–103.

Imielinski, T. and Badrinath, B. (1994). Mobile wireless computing: challenges in data management. *Communications of the ACM*, 37(10):18–28.

Jain, R., Chiu, D.-M., and Hawe, W. R. (1984). *A quantitative measure of fairness and discrimination for resource allocation in shared computer system*, volume 38. Eastern Research Laboratory, Digital Equipment Corporation Hudson, MA.

Jang, S. H. and Lee, J. S. (2011). Mobile resource reliability-based job scheduling for mobile grid. *KSII Transactions on Internet & Information Systems*, 5(1).

Jonas, E., Venkataraman, S., Stoica, I., and Recht, B. (2017). Occupy the cloud: Distributed computing for the 99%. *arXiv preprint arXiv:1702.04024*.

Katsaros, K. and Polyzos, G. C. (2007). Optimizing operation of a hierarchical campus-wide mobile grid for intermittent wireless connectivity. In *2007 15th IEEE Workshop on Local Metropolitan Area Networks*, pages 111–116.

Kesaraju, V. and Ciarallo, F. W. (2012). Integrated simulation combining process-driven and event-driven models. *Journal of Simulation*, 6(1):9–20.

Khalaj, A., Lutfiyya, H., and Perry, M. (2010). The proxy-based mobile grid. In Cai, Y., Magedanz, T., Li, M., Xia, J., and Giannelli, C., editors, *Mobile Wireless Middleware, Operating Systems, and Applications*, volume 48 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 59–69. Springer Berlin Heidelberg.

Khan, A. U. R., Othman, M., Xia, F., and Khan, A. N. (2015). Context-aware mobile cloud computing and its challenges. *IEEE Cloud Computing*, 2(3):42–49.

Khan, W. Z., Xiang, Y., Aalsalem, M. Y., and Arshad, Q. (2013). Mobile phone sensing systems: A survey. *IEEE Communications Surveys & Tutorials*, 15(1):402–427.

Kliazovich, D., Bouvry, P., and Khan, S. U. (2012). Greencloud: a packet-level simulator of energy-aware cloud computing data centers. *The Journal of Supercomputing*, 62(3):1263–1283.

Kotz, D. and Essien, K. (2005). Analysis of a campus-wide wireless network. *Wireless Networks*, 11(1-2):115–133.

Kotz, D., Henderson, T., Abyzov, I., and Yeo, J. (2009). CRAWDAD dataset dartmouth/campus (v. 2009-09-09). Downloaded from http://crawdad.org/dartmouth/campus/20090909.

Krauter, K., Buyya, R., and Maheswaran, M. (2002). A taxonomy and survey of grid resource management systems for distributed computing. *Software: Practice and Experience*, 32(2):135–164.

Lee, D., Lee, H., Park, D., and Jeong, Y.-S. (2013). Proxy based seamless connection management method in mobile cloud computing. *Cluster Computing*, 16(4):733–744.

Lee, J., Choi, S., Gil, J., Suh, T., and Yu, H. (2014). A scheduling algorithm with dynamic properties in mobile grid. *Frontiers of Computer Science*, 8(5):847–857.

Li, B., Pei, Y., Wu, H., and Shen, B. (2015). Heuristics to allocate high-performance cloudlets for computation offloading in mobile ad hoc clouds. *The Journal of Supercomputing*, 71(8):3009–3036.

Litke, A., Skoutas, D., and Varvarigou, T. (2004). Mobile grid computing: Changes and challenges of resource management in a mobile grid environment. In *5th International Conference on Practical Aspects of Knowledge Management (PAKM 2004)*.

Liu, J., Ahmed, E., Shiraz, M., Gani, A., Buyya, R., and Qureshi, A. (2015). Application partitioning algorithms in mobile cloud computing: Taxonomy, review and future directions. *Journal of Network and Computer Applications*, 48:99–117.

Liu, M., Li, Z., Guo, X., and Dutkiewicz, E. (2008). Performance analysis and optimization of handoff algorithms in heterogeneous wireless networks. *IEEE Transactions on Mobile Computing*, 7(7):846–857.

Loke, S. W., Napier, K., Alali, A., Fernando, N., and Rahayu, W. (2015). Mobile computations with surrounding devices: Proximity sensing and multilayered work stealing. *ACM Transactions on Embedded Computing Systems*, 14(2):22:1–22:25.

Macone, D., Oddi, G., and Pietrabissa, A. (2013). Mq-routing: Mobility-, gps- and energy-aware routing protocol in {MANETs} for disaster relief scenarios. *Ad Hoc Networks*, 11(3):861 – 878.

Makris, P., Skoutas, D. N., and Skianis, C. (2013). A survey on context-aware mobile and wireless networking: On networking and computing environments' integration. *IEEE communications surveys & tutorials*, 15(1):362–386.

Marinescu, D. C., Marinescu, G. M., Ji, Y., Boloni, L., and Siegel, H. J. (2003). Ad hoc grids: communication and computing in a power constrained environment. In *Conference Proceedings of the 2003 IEEE International Performance, Computing, and Communications Conference, 2003.*, pages 113–122.

Mednieks, Z., Dornin, L., Meike, G. B., and Nakamura, M. (2012). *Programming Android, 2nd Edition, Java Programming for the New Generation of Mobile Devices*. O'Reilly Media, 2 edition.

Miluzzo, E., Cáceres, R., and Chen, Y.-F. (2012). Vision: mclouds-computing on clouds of mobile devices. In *Proceedings of the third ACM workshop on Mobile cloud computing and services*, pages 9–14. ACM.

Monares, Á., Ochoa, S. F., Pino, J. A., Herskovic, V., Rodriguez-Covili, J., and Neyem, A. (2011). Mobile computing in urban emergency situations: Improving the support to firefighters in the field. *Expert systems with applications*, 38(2):1255–1267.

Mtibaa, A., Fahim, A., Harras, K. A., and Ammar, M. H. (2013). Towards resource sharing in mobile device clouds: Power balancing across mobile devices. In *ACM SIGCOMM Computer Communication Review*, volume 43, pages 51–56. ACM.

Murray, D. G., Yoneki, E., Crowcroft, J., and Hand, S. (2010). The case for crowd computing. In *Proceedings of the second ACM SIGCOMM workshop on Networking, systems, and applications on mobile handhelds*, pages 39–44. ACM.

Park, S.-M., Ko, Y.-B., and Kim, J.-H. (2003). Disconnected operation service in mobile grid computing. In *Service-Oriented Computing-ICSOC 2003*, pages 499–513. Springer.

Pasupuleti, S. K., Ramalingam, S., and Buyya, R. (2016). An efficient and secure privacy-preserving approach for outsourced data of resource constrained mobile devices in cloud computing. *Journal of Network and Computer Applications*, 64:12–22.

Pejovic, V. and Musolesi, M. (2015). Anticipatory mobile computing: A survey of the state of the art and research challenges. *ACM Computing Surveys (CSUR)*, 47(3):47.

Perez Campos, A., Rodriguez, J. M., and A., Z. (2017). An empirical evaluation of a simple energy aware scheduler for mobile grids. In *XLIII CLEI - Inpress*, page in press. IEEE.

Pérez-Torres, R., Torres-Huitzil, C., and Galeana-Zapién, H. (2016). Power management techniques in smartphone-based mobility sensing systems: A survey. *Pervasive and Mobile Computing*.

Rahmati, A., Qian, A., and Zhong, L. (2007). Understanding human-battery interaction on mobile phones. In *Proceedings of the 9th International Conference on Human Computer Interaction with Mobile Devices and Services*, MobileHCI '07, pages 265–272, New York, NY, USA. ACM.

Rajovic, N., Carpenter, P. M., Gelado, I., Puzovic, N., Ramirez, A., and Valero, M. (2013). Supercomputing with commodity cpus: Are mobile socs ready for hpc? In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, page 40. ACM.

Rego, P. A., Costa, P. B., Coutinho, E. F., Rocha, L. S., Trinta, F. A., and de Souza, J. N. (2016). Performing computation offloading on multiple platforms. *Computer Communications*.

Restuccia, F., Das, S. K., and Payton, J. (2016). Incentive mechanisms for participatory sensing: Survey and research challenges. *ACM Transactions on Sensor Networks (TOSN)*, 12(2):13.

Rezende, C., Boukerche, A., Pazzi, R. W., Rocha, B. P., and Loureiro, A. A. (2011). The impact of mobility on mobile ad hoc networks through the perspective of complex networks. *Journal of Parallel and Distributed Computing*, 71(9):1189–1200.

Rice, A. and Hay, S. (2010). Measuring Mobile Phone Energy Consumption for 802.11 Wireless Networking. *Pervasive and Mobile Computing*, 6(6):593–606.

Robinson, S. (2014). *Simulation: the practice of model development and use*. Palgrave Macmillan.

Rodriguez, J. M., Mateos, C., and Zunino, A. (2012). Are smartphones really useful for scientific computing? *Lecture Notes In Computer Science*, 7547:38–47.

Rodriguez, J. M., Mateos, C., and Zunino, A. (2014). Energy-efficient job stealing for cpu-intensive processing in mobile devices. *Computing*, 96(2):87–117.

Rodriguez, J. M., Zunino, A., and Campo, M. (2010). Mobile grid seas: Simple energy-aware scheduler. In *3rd High-Performance Computing Symposium - 39th JAIIO*.

Rodriguez, J. M., Zunino, A., and Campo, M. (2011). Introducing mobile devices into grid systems: a survey. *International Journal of Web and Grid Services*, 7(1):1–40.

Satyanarayanan, M. (2001). Pervasive computing: Vision and challenges. *IEEE Personal communications*, 8(4):10–17.

Satyanarayanan, M. (2010). Mobile computing: The next decade. In *Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond*, MCS '10, pages 5:1–5:6, New York, NY, USA. ACM.

Shah, S. C. (2015). Energy efficient and robust allocation of interdependent tasks on mobile ad hoc computational grid. *Concurrency and Computation: Practice and Experience*, 27(5):1226–1254.

Shah, S. C., Chauhdary, S. H., Park, M.-S., et al. (2012). An effective and robust two-phase resource allocation scheme for interdependent tasks in mobile ad hoc computational grids. *Journal of Parallel and Distributed Computing*, 72(12):1664–1679.

Sharifi, M., Kafaie, S., and Kashefi, O. (2012). A survey and taxonomy of cyber foraging of mobile devices. *IEEE Communications Surveys Tutorials*, 14(4):1232–1243.

Shi, C., Lakafosis, V., Ammar, M. H., and Zegura, E. W. (2012). Serendipity: enabling remote computing among intermittently connected mobile devices. In *Proceedings of the thirteenth ACM international symposium on Mobile Ad Hoc Networking and Computing*, pages 145–154. ACM.

Shi, T., Yang, M., Li, X., Lei, Q., and Jiang, Y. (2016). An energy-efficient scheduling scheme for time-constrained tasks in local mobile clouds. *Pervasive and Mobile Computing*, 27:90–105.

Shivle, S., Siegel, H., Maciejewski, A. A., Sugavanam, P., Banka, T., Castain, R., Chindam, K., Dussinger, S., Pichumani, P., Satyasekaran, P., Saylor, W., Sendek, D., Sousa, J., Sridharan, J.,

and Velazco, J. (2006). Static allocation of resources to communicating subtasks in a heterogeneous ad hoc grid environment. *Journal of Parallel and Distributed Computing*, 66(4):600 – 611. Algorithms for Wireless and Ad-Hoc Networks.

Singh, K. V. and Raza, Z. (2017). A quantum-inspired binary gravitational search algorithm–based job-scheduling model for mobile computational grid. *Concurrency and Computation: Practice and Experience*, 29(12).

Sokolowski, J. A. and Banks, C. M. (2010). *Modeling and simulation fundamentals: theoretical underpinnings and practical domains*. John Wiley & Sons.

Spano, D. and Ricciato, F. (2016). Opportunistic time-of-arrival localization in fully asynchronous wireless networks. *Pervasive and Mobile Computing*.

Spillner, J., Mateos, C., and Monge, D. A. (2017). Faaster, better, cheaper: The prospect of serverless scientific computing and hpc. In *4th Latin American Conference on High Performance Computing (CARLA). To appear*.

Takacs, G., Choubassi, M. E., Wu, Y., and Kozintsev, I. (2011). 3d mobile augmented reality in urban scenes. In *2011 IEEE International Conference on Multimedia and Expo*, pages 1–4.

Takeno, K., Ichimura, M., Takano, K., and Yamaki, J. (2005). Influence of cycle capacity deterioration and storage capacity deterioration on li-ion batteries used in mobile phones. *Journal of Power Sources*, 142(1-2):298–305.

Topcuoglu, H., Hariri, S., and Wu, M.-y. (2002). Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE transactions on parallel and distributed systems*, 13(3):260–274.

Urbah, E., Kacsuk, P., Farkas, Z., Fedak, G., Kecskemeti, G., Lodygensky, O., Marosi, A., Balaton, Z., Caillat, G., Gombas, G., et al. (2009). Edges: Bridging egee to boinc and xtremweb. *Journal of Grid Computing*, 7(3):335–354.

Vaithiya, S. S. and Bhanu, S. M. S. (2012). Zone based job scheduling in mobile grid environment. *International Journal of Grid Computing & Applications (IJGCA)*, 3(2).

van Nieuwpoort, R., Wrzesinska, G., Jacobs, C. J. H., and Bal, H. E. (2010). Satin: A high-level and efficient grid programming model. *ACM Trans. Program. Lang. Syst.*, 32(3).

Varghese, B. and Buyya, R. (2017). Next generation cloud computing: New trends and research directions. *Future Generation Computer Systems*.

Vega, D., Meseguer, R., Freitag, F., and Ochoa, S. F. (2013). Effort-based incentives for resource sharing in collaborative volunteer applications. In *Computer Supported Cooperative Work in Design (CSCWD), 2013 IEEE 17th International Conference on*, pages 37–42. IEEE.

Viswanathan, H., Lee, E. K., Rodero, I., and Pompili, D. (2015). Uncertainty-aware autonomic resource provisioning for mobile cloud computing. *IEEE transactions on parallel and distributed systems*, 26(8):2363–2372.

Wagner, D. T., Rice, A., and Beresford, A. R. (2014). Device analyzer: Large-scale mobile data collection. *SIGMETRICS Perform. Eval. Rev.*, 41(4):53–56.

Wei, X., Fan, J., Lu, Z., and Ding, K. (2013). Application scheduling in mobile cloud computing with load balancing. *Journal of Applied Mathematics*, 2013.

Weiser, M. (1993). Ubiquitous computing. *Computer*, 26(10):71–72.

Wessel, S., Huber, M., Stumpf, F., and Eckert, C. (2015). Improving mobile device security with operating system-level virtualization. *Computers & Security*, 52:207–220.

Wilhelm, R., Engblom, J., Ermedahl, A., Holsti, N., Thesing, S., Whalley, D., Bernat, G., Ferdinand, C., Heckmann, R., Mitra, T., Mueller, F., Puaut, I., Puschner, P., Staschulat, J., and Stenström, P. (2008). The worst-case execution-time problem - overview of methods and survey of tools. *ACM Transactions on Embedded Computing Systems*, 7(3):36:1–36:53.

Yang, Y., Qiu, X.-s., Meng, L.-m., and Rui, L.-l. (2010). A self-adaptive method of task allocation in clustering-based manets. In *Network Operations and Management Symposium (NOMS), 2010 IEEE*, pages 440–447. IEEE.

Yi, S.-Y. and Livny, M. (1999). Extending the condor distributed systems for mobile clients. *ACM SIGMOBILE Mobile Computing and Communications Review*, 3(1):38–46.

Yick, J., Mukherjee, B., and Ghosal, D. (2008). Wireless sensor network survey. *Computer networks*, 52(12):2292–2330.

Yousafzai, A., Gani, A., Noor, R. M., Naveed, A., Ahmad, R. W., and Chang, V. (2016). Computational offloading mechanism for native and android runtime based mobile applications. *Journal of Systems and Software*, 121:28–39.

Zhang, J., Huang, H., and Wang, X. (2016). Resource provision algorithms in cloud computing: A survey. *Journal of Network and Computer Applications*, 64:23–42.

Zola, E. and Kassler, A. (2016). Minimizing the impact of the handover for mobile users in wlan: A study on performance optimization. *Computer Networks*, pages –.