Uncertainty-Aware Autonomic Resource Provisioning for Mobile Cloud Computing

Hariharasudhan Viswanathan, *Student Member, IEEE*, Eun Kyung Lee, *Student Member, IEEE*, Ivan Rodero, *Member, IEEE*, and Dario Pompili, *Senior Member, IEEE*

Abstract—Mobile platforms are becoming the predominant medium of access to Internet services due to the tremendous increase in their computation and communication capabilities. However, enabling applications that require real-time in-the-field data collection and processing using mobile platforms is still challenging due to i) the insufficient computing capabilities and unavailability of complete data on individual mobile devices and ii) the prohibitive communication cost and response time involved in offloading data to remote computing resources such as cloud datacenters for centralized computation. A novel resource provisioning framework for organizing the heterogeneous sensing, computing, and communication capabilities of static and mobile devices in the vicinity in order to form an elastic resource pool—a hybrid static/mobile computing grid (also called a loosely-coupled mobile device cloud)—is presented. This local computing grid can be harnessed to enable innovative data- and compute-intensive mobile applications such as ubiquitous context-aware health and wellness monitoring of the elderly, distributed rainfall and flood-risk estimation, distributed object recognition and tracking, and content-based distributed multimedia search and sharing. In order to address challenges such as the inherent uncertainty in the hybrid grid (in terms of network connectivity and device availability), the proposed role-based resource provisioning framework is imparted with autonomic capabilities, namely, self-organization, self-optimization, and self-healing. A thorough experimental analysis aimed at verifying and demonstrating the benefits brought by autonomic capabilities of the framework is also presented in detail.

Index Terms—Autonomics, uncertainty, mobile clouds, mobile grids, self-organization, self-optimization, self-healing

1 Introduction

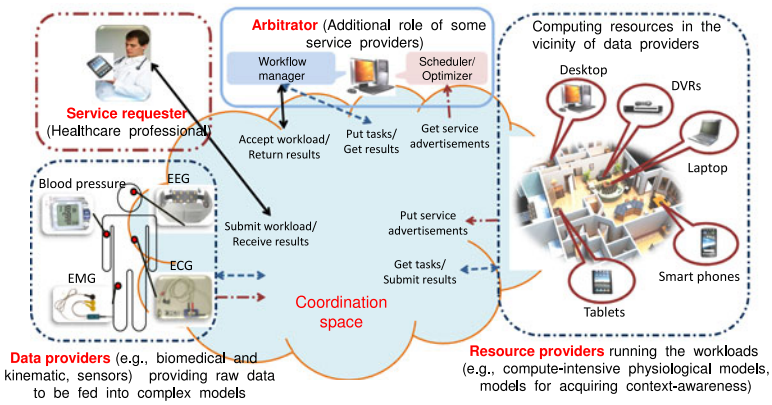
The computation and communication capabilities of mobile devices such as smart phones, tablets, netbooks, and laptops have improved tremendously due to the advances in microprocessor, storage, and wireless technologies. It has been projected that, by 2015, mobile devices will surpass wired devices as the most preferred medium of access to the Internet. These changes will have a significant impact on the underlying resource pool in distributed computing paradigms that use Internet-connected devices, volunteered by their owners, as a source of computing power and storage [1]. Also, as more and more of these mobile devices are coupled with in-built as well as external sensors capable of monitoring ambient conditions, acceleration, orientation, gravity, etc., and Global Positioning System (GPS) receivers, they can provide spatially distributed measurements regarding the environment in their proximity. In addition, advances in the field of wireless sensors has led to the development of compact sensor nodes capable of communicating with other mobile devices and of capturing a wide variety of sensor data—from biomedical (e.g., electrocardiogram, galvanic skin response, oxygen saturation) to kinematic (e.g., acceleration, angular velocity).

This paper presents a resource provisioning framework for organizing the heterogeneous sensing, computing, and communication capabilities of static and mobile devices in the vicinity in order to form an elastic resource pool—*a hybrid static/mobile computing grid*. In literature, such a resource pool is also referred to as *a "loosely-coupled" mobile device cloud* and hence, in this paper, we use the terms *mobile cloud* and *mobile grid* interchangeably. This local computing grid can be harnessed to enable innovative *data-* and *compute-intensive distributed mobile applications* such as ubiquitous context-aware health and wellness monitoring of the elderly, rainfall and flood-risk estimation, estimation of pollution level using real-time air-quality measurements, object recognition and tracking, and content-based multimedia search and sharing. The response time, quality, and relevance of such mobile applications, which rely on *real-time in-the-field processing of locally generated data*, can be drastically improved using our envisioned framework. Currently, the primary impediments to real-time in-the-field data processing are, 1) insufficient sensing and computing capabilities on individual mobile devices, which prevents them from producing meaningful results within realistic time bounds *in isolation*, and 2) the prohibitive communication cost and response time involved in enabling such data-intensive applications using the *wired-grid-computing* and/or *cloud-computing* approaches alone [2]—in which computation and storage are fully offloaded to remote computing resources on the Internet.

Resource provisioning in the aforementioned computing grid is a challenging problem due to inherent *uncertainty* in terms of network connectivity and device availability. This uncertainty can be attributed to unpredictable node mobility, varying rate of battery drain, susceptibility to hardware

• *The authors are with the Rutgers Discovery Informatics Institute (RDI²) and the NSF Center for Cloud and Autonomic Computing (CAC), Department of Electrical and Computer Engineering, Rutgers University— New Brunswick, NJ. E-mail: {hari_viswanathan, eunkyung_lee, irodero, pompili}@cac.rutgers.edu.*

Manuscript received 15 Nov. 2013; revised 18 May 2014; accepted 14 July 2014. Date of publication 30 July 2014; date of current version 6 July 2015. Recommended for acceptance by M.-C. Chuah.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPDS.2014.2345057



(a)

FOCUS	Chu et al. [3]	Giurgiu et al. [4]	Chun et al. [5]	Darby et al. [6]	Costa et al. [7]	Huang et al. [8]	Lima et al. [9]	Jiang et al. [10]	Liao et al. [11]	OURS
Infrastructure										
Mobile					✓		✓	✓	✓	
Static						✓				
Hybrid	✓	✓	✓	✓						✓
Concern										
Energy			✓			✓				✓
Connectivity				✓	✓					✓
Uncertainty				✓						✓
Contribution										
Protocol				✓			✓		✓	
Middleware	✓	✓	✓		✓	✓		✓		✓

(b)

Fig. 1. (a) Autonomic resource provisioning enabling ubiquitous healthcare; (b) summary of related work.

failures, and lack of a priori knowledge about the application performance on different mobile hardware and software platforms. In order to address the research challenges associated with reliable hybrid grid coordination and application performance (in terms of response time) under uncertainty, we impart our proposed framework with autonomic capabilities—*self-organization*, *self-optimization*, and *self-healing*. Applications are made up of one or more workloads, which is usually composed of multiple tasks whose order of execution is specified by a workflow. Tasks here may refer to data-processing and/or data-analysis operations with different computational, storage, and deadline requirements. Applications may exhibit *data parallelism* (in which data is distributed across different parallel computing nodes that perform the same task) or exhibit *task parallelism* (in which parallel computing nodes may perform different tasks on the same or different data).

In our solution, the entities of the hybrid grid may at any time play one or more of the following three *logical roles* as shown in Fig. 1a: i) *service requester*, which places requests for workloads that require additional data and/or computing resources from other devices, ii) *Service Provider*, which can be a *Data Provider (DP)*, *Resource Provider (RP)*, or both, and iii) *arbitrator* (also typically known as broker), which processes the requests from the requesters, determines the set of service providers that will provide or process data, and distributes the workload tasks among them. Data providers provide scalar or multimedia data while resource providers lend their computational (CPU cycles), storage (volatile and non-volatile memory), and communication (i.e., network interface capacity) resources for processing data. The arbitrator—an additional role played by some of the service providers—is aided by a novel *uncertainty- and energy-aware resource allocation engine*, which will distribute the workload tasks among the service providers. This way, we ensure that the data providers do not drain valuable energy and, in turn, maximize their lifetime as the sensor data that they provide is crucial for enabling data-driven mobile applications.

Related work: Fig. 1a depicts the entities of a hybrid grid enabling an ubiquitous healthcare application that relies on processing collected biomedical data in-the-field for real-time physiological monitoring. Prior research efforts—specifically, in the field of mobile grid computing—aimed at integrating mobile devices into the wired-grid [3], [7], [9] and cloud-computing infrastructure [4], [5], [12], [13] mainly as service requesters. On the other hand, research efforts in the field of delay-tolerant distributed computing has led to the development of a new paradigm called Opportunistic Computing (OC) [14], [15], which depends entirely on direct encounters to opportunistically share data and computing resources. Differently from these two efforts, we organize the mobile devices into a heterogeneous resource pool (with entities playing one or more logical roles—arbitrators, service requesters, service providers) and also address uncertainty-aware resource management for ensuring application Quality of Service (QoS) even in highly dynamic and unpredictable environments. In the table depicted in Fig. 1b, we summarize previous work in the field of mobile grid computing and position our uncertainty-aware resource provisioning framework for real-time in-the-field data processing in heterogeneous mobile computing grids. In Appendix A of the supplementary material, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2014.2345057, we discuss the state of the art in management of mobile grids in detail and motivate the need for our approach.

Our resource allocation engine leverages *long-term statistics* regarding the dynamics of the underlying resource pool and utilizes the novel concept of *application waypoints* to monitor continuously the effect of the aforementioned uncertainties on application performance. Specifically, the long-term statistics are exploited to minimize the effect of uncertainty arising from inaccurate estimates of device availability (due to unpredictable mobility), while the application waypoints are used to minimize the effect of uncertainty arising out of inaccurate estimates of application performance (due to unpredictable battery drain and resource utilization). Waypoints also impart the desired robustness under device failures and help us eliminate assumptions such as existence of accurate models for application performance on different mobile hardware and software platforms. Our major contributions include:

- A role-based and uncertainty-aware architectural framework for imparting the self-organization

capability, i.e., for handling service discovery and service request arrivals as well as for task distribution and management.
- A novel energy- and uncertainty-aware resource allocation engine for imparting self-optimization and self-healing properties, i.e., for allocating the workload tasks optimally among the computing devices and to ensure application QoS even under uncertainties.
- A thorough performance analysis of our resource provisioning framework for mobile grids through experiments on a prototype testbed of Android- and Linux-based mobile devices as well as simulations.

This paper is an extended version of [16] in which we discuss only a subset of the autonomic capabilities. The remainder of this paper is organized as follows. In Section 2, we present our autonomic resource provisioning framework for mobile grids and provide details on how we impart the self-organizing, self-optimizing, and self-healing properties. In Section 3, we describe our experimental methodology and results. Finally, in Section 4, we present our conclusions.

2 Proposed Solution

Our contributions are geared towards imparting autonomic capabilities to the resource management framework. Mechanisms for service discovery and workload management will impart the self-organization capability. The energy-aware resource provisioning engine will impart self-optimization, while the uncertainty handling mechanism will bestow the self-healing capability.

2.1 Role-Based Architectural Framework

The self-organization capability (for handling service discovery and service request arrivals as well as for task distribution and management) is imparted by the role-based architectural framework. It also facilitates interactions among the mobile entities for coordination and seamless switching among the three logical roles, namely, *service requesters*, *service providers*, and *arbitrators*.

Service discovery: Service discovery at the arbitrators is achieved through voluntary service advertisements from the service providers. Service advertisements will include information about the current position, amount of computing (γ_n^{cpu}, in terms of unutilized CPU cycles [percent]), memory (γ_n^{mem} [Bytes]), and communication (γ_n^{net} [bps]) resources, the start (t_n^{in}) and end (t_n^{out}) times of the availability of those resources, and the available battery capacity (e_n^{adv} [Wh]) at each SP n. The arbitrator is aware of the instantaneous power drawn by the workload tasks of a specific application when running on a specific class of CPU and memory (together given by c_n^{comp} [W]) as well as network (c_n^{net} [W]) resources at each SP as the information about the different types of devices is known in advance. Even though a random arbitrator assignment can be adopted, we advocate the use of a distributed self-election mechanism for assigning the appropriate number of arbitrators. Details of the same are available in Appendix B of the supplementary material, available online.

Workload management: Each arbitrator is composed of two components, namely, *workload manager* and *scheduler/optimizer*, as shown in the top of Fig. 1a. The workload manager tracks workload requests, allocates workload tasks among service providers, and aggregates results. The optimizer identifies the number of service providers available for the requested duration and determines the optimal distribution of workload tasks among them. *In this paper, we focus on uncertainty-aware task allocation for data-parallel applications only*. For a discussion on task allocation for task-parallel applications refer to our contributions in [17]. Data-parallel applications are also referred to as "embarrassingly parallel" (map-reduce-type) applications in which the independent set of tasks (either homogeneous or heterogeneous) can be performed in parallel. The results of the tasks are fused for generating the final result.

The optimizer shares the workload submitted by the data providers among the available service providers based on one of several possible policies. The different tasks of a workload may be distributed among the available service providers based on an energy-aware policy that aims at minimizing the maximum battery drain. This can be achieved through minimization of computational load on each individual service provider by exploiting parallelism while incurring a very low communication cost. Another policy may just place emphasis on minimizing the response time without considering battery drain.

2.2 Energy-Aware Resource Allocation

Here, we explain our resource allocation engine, which implements one of the aforementioned policies (energy aware) for hybrid grids in detail. The following is the sequence of events happening at one of the arbitrators. Similar events happen simultaneously at the other arbitrators in the computing grid. When a service requester needs additional data or computing resources, it submits a service request to the nearest arbitrator and also specifies δ^{max} [s], the maximum duration for which it is ready to wait for a service response. The arbitrator extracts the following information based on the service advertisements: the devices' (service providers') capability, $\overline{\Gamma}^x = \{\gamma_n^x\}_{1\times N}$, where $x = cpu, mem, net$; the associated costs, $\overline{C}^{comp} = \{c_n^{comp}\}_{1\times N}$ and $\overline{C}^{net} = \{c_n^{net}\}_{1\times N}$; the devices' availability, $\overline{T}^{in} = \{t_n^{in}\}_{1\times N}$ and $\overline{T}^{out} = \{t_n^{out}\}_{1\times N}$; and their battery status $\overline{E}^{adv} = \{e_n^{adv}\}_{1\times N}$. The variables that the optimization problem has to find are

$$\textbf{Find: } \mathbf{A}, \overline{\Delta}^d, \overline{\Delta}^s. \tag{1}$$

Here, $\mathbf{A} = \{a_{ij}\}_{N\times N}$ conveys the associativity of data provider i with SP j, which is determined by the arbitrator, $\overline{\Delta}^d = \{\delta_n^d\}_{1\times N}$ [h] conveys the duration for which the services of each SP will be used for data collection, and $\overline{\Delta}^s = \{\delta_n^s\}_{1\times N}$ [h] conveys the duration for which the resources of each SP will be used for computation (cpu, mem, and net) and/or for multi-hop communication (net) as a relay node. In this formulation, the objective of the optimization problem in (2) is the *maximization of minimum residual battery capacity* at all the SPs, $\max \min_n e_n^{res}$—where e_n^{res} [Wh] is computed as in (3)—while ensuring that the service response is delivered within δ^{max}. This objective maximizes the lifetime of every single SP and, thus, maintains the heterogeneity of

the resource pool for longer periods. The set of SPs and the duration for which each of their capabilities are availed will be determined by considering the tradeoffs among the cost (in terms of battery drain) e_n^{data} [Wh] (4) for transferring the data locally from data providers to the resource providers, the computational cost e_n^{comp} [Wh] (5) for availing the computational capabilities of the resource providers for servicing the request and for aggregating and generating the result.

$$\textbf{Maximize:} \quad \min_n e_n^{res}, \tag{2}$$

$$where, \quad e_n^{res} = e_n^{adv} - \left(e_n^{data} + e_n^{comp}\right); \tag{3}$$

$$e_n^{data} = \delta_n^d \cdot c_n^{net}; \tag{4}$$

$$e_n^{comp} = u_n \cdot \delta_n^s \cdot c_n^{comp}. \tag{5}$$

In (3), $e_n^{data} + e_n^{comp}$ is the amount of battery capacity drained at each service provider n; δ_n^d for a service provider n depends on the amount of data it has to transmit (ω [Bytes] as a data provider) or aggregate ($\omega \cdot \sum_{i=1}^{N} a_{in}$ [Bytes] as a resource provider), and the availed communication capability, given by,

$$\delta_n^d = \begin{cases} f\left(\omega, \gamma_n^{net}\right) & \text{if } u_n = 0, \\ f\left(\omega \cdot \sum_{i=1}^{N} a_{in}, \gamma_n^{net}\right) & \text{if } u_n = 1. \end{cases} \tag{6}$$

For simplicity, ω is considered to be the problem size of a trivial task and each data provider provides the same amount of data. However, this is easily generalizable to a case where each data provider provides a different amount of data, in which case the problem sizes of each trivial task will be different. Function $f()$ monotonically increases as the amount of data to be transmitted or received increases; δ_n^s for a service provider n depends on the amount of data it has to process and the availed computing capabilities specified by γ_n^{cpu} and γ_n^{mem}, given by,

$$\delta_n^s = g\left(\gamma_n^{cpu}, \gamma_n^{mem}, \omega \cdot \sum_{i=1}^{N} a_{in}\right). \tag{7}$$

Function $g()$ monotonically increases with the amount of data to be processed. The *constraints* to the optimization problem are, $\forall n = 1 \ldots N$,

$$s_n \geq u_n; \quad 0 \leq \delta_n^d, \delta_n^s; \tag{8}$$

$$\delta_n^s \leq \min\left\{t_n^{out}, t^{now} + \delta^{max}\right\} - \max\left\{t^{now} + \delta_n^d, t_n^{in}\right\}; \tag{9}$$

$$\delta_n^d \cdot c_n^{net} + u_n \cdot \delta_n^s \cdot c_n^{comp} \leq e_n^{adv}. \tag{10}$$

Constraint (8) ensures that only a resource provider is chosen to perform the computing. Constraint (9) ensures that the consumer's deadline for service response is met while also utilizing a service provider only for the duration for which its services are advertised to be available. Constraint (10) ensures that the battery capacity is not exceeded.

2.3 Uncertainty Awareness

The resource allocation engine of our proposed in-situ data processing system is capable of handling uncertainties in the highly dynamic hybrid heterogeneous computing environment. We identify the different sources of uncertainties and bestow the resource allocation engine with the desired properties to guarantee application QoS (in terms of response time) even under those uncertainties.

Sources of uncertainty: *Inaccurate estimation of the availability (duration) of SP* is a major source of uncertainty that results in a large number of incomplete workload task migrations. The duration of availability specified in the service advertisements is based on the battery drain estimates and may not accurately reflect the duration for which the service provider will be associated with the arbitrator. One or more of the SPs may lose network connectivity to the arbitrator or go offline.

Inaccuracy in the estimation of task completion times—function $g()$ in the aforementioned optimization problem—is one of the sources of uncertainties that affect application QoS. This is especially true when the behavior of a workload task (execution time and resource utilization) is not known in advance at the arbitrator. The uncertainty can be reduced to a certain extent in our application scenario by profiling the behavior of the workload in advance. *However, some models exhibit radically different behaviors depending on the type of inputs (e.g., sorted/unsorted, dense/sparse).*

Uncertainty in workload completion within the response-time bound arises when a SP is experiencing an unexpected *increase in the rate of battery drain* (and runs out of energy) due to any of the additional critical operations that it may be performing at the same time as the workload task. The optimization problem may also *over- or under-provision* computing resources due to an inaccurate estimate of task completion time. The former would result in unnecessary wastage of energy (battery drain) while the latter would result in violation of QoS. While the arbitrator can be made aware of any problems at the SP using feedback, handling situations such as loss of network connectivity and hardware failures is a challenging task.

Uncertainty-aware self-organization: In order to impart the uncertainty-aware self-organization capability to the proposed resource-allocation framework, we designed mechanisms that help the arbitrator extract the following long-term statistics from the underlying resource pool: the average arrival (joining) rate of SPs ($\widetilde{W}$), the average SP availability duration ($\widetilde{T}$), and the average number of SPs associated with the arbitrator at any point in time ($\widetilde{N}$). The relationship among these three long-term statistics is given by Little's law [18], $\widetilde{N} = \widetilde{W} \cdot \widetilde{T}$. The arbitrators update continuously these statistics and share at least two of the three aforementioned averages with its successors if and when an arbitrator's handoff happens. Knowledge of these average statistics helps the arbitrators assess the *churn rate* of SPs. Churn rate is a measure of the number of service providers moving into or out of an arbitrator's resource pool over a specific period of time. Note that the arbitrators need not extract or be aware of the underlying probability distribution of service provider arrivals or of availability durations. The long-term averages, which are easy to acquire and maintain, are sufficient.

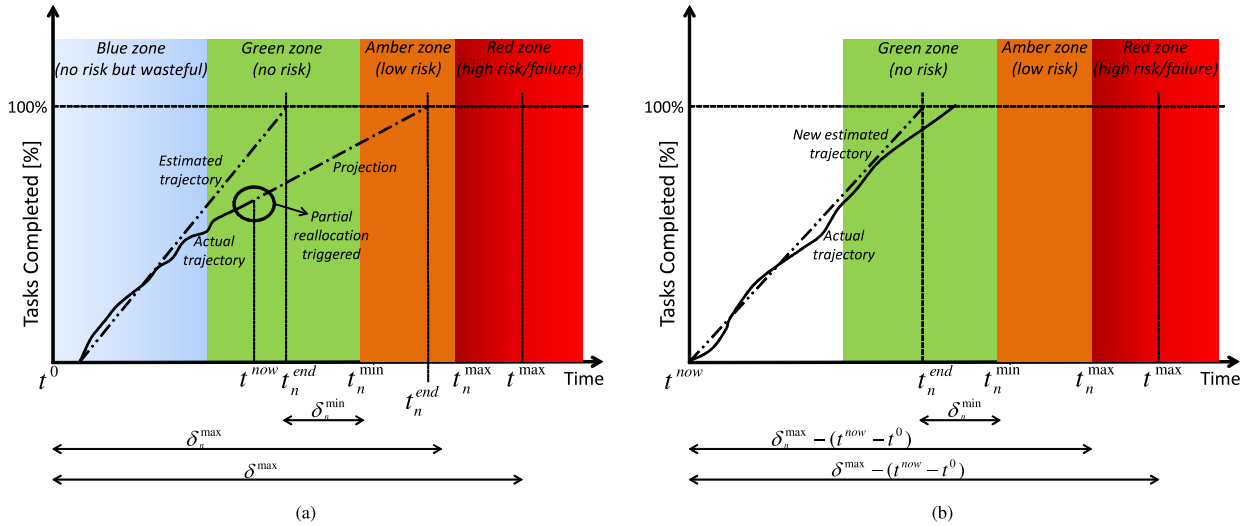


Fig. 2. Illustration of the use of *application waypoints* to report progress of workload task completion at a service provider n. (a) shows the estimated trajectory of task completion and the relative position of the estimated completion time t_n^{end} with respect to the four end zones Blue, Green, Amber, and Red. The projection at time t^{now} falls in the Amber zone triggering a partial reallocation of tasks from this troubled service provider; (b) shows the new estimated and the actual trajectory of task completion at the troubled service provider after it has been relieved of some tasks. The SP is able to complete its allocated tasks with the service providers' deadline t_n^{max}. If the projection had fallen in the Red zone, it would have triggered a full reallocation, i.e., all the incomplete tasks at the service provider.

Churn rate of SPs will be different in different geographic location. For example, the churn rate of SPs at a shopping mall is far greater than the one at a coffee shop. Also, at a particular location (say, the coffee shop), the churn rate can vary over time (e.g., depending on the time of the day). When the churn rate is high, i.e., the average duration of SP availability is low, the percentage of potentially costly migrated workload tasks will be high if the resource-allocation engine does not possess uncertainty awareness. When the long-term average of availability duration is not taken into account at the arbitrator and when the durations advertised by the SPs are used as constraints in the resource allocation problem, it results in a mismatch between the ground reality and the optimization at the arbitrator. However, our framework with uncertainty awareness achieves a smooth degradation (if any) in QoS (because of the small number of task migrations) when churn rate increases as it effectively exploits the knowledge gathered over time.

2.4 Uncertainty Handling

To ensure application QoS under all of the aforementioned circumstances, we introduce the novel idea of *application waypoints*, at which the SPs report to the arbitrator with intermediate results and their progress. Waypoints enable the arbitrators to estimate the *residual tasks' execution time* for each SP. If an arbitrator does not receive any waypoints from a SP, it marks that SP as failed after a timeout and assigns the incomplete tasks to one or more backup SPs. If an arbitrator receives waypoints from a SP at a lower rate than the expected (from offline application profiling detailed in Appendix C in the supplementary file, available online), it has to determine *when* and *how* to intervene, i.e., when to relieve the slow service provider of some tasks and which SPs to use as backup. The details of our proposed reactive measurement-based self-healing mechanism for heterogeneous mobile computing grids follow.

Application waypoints: Once the arbitrator assigns sets of tasks to the different SPs, it continuously tracks their progress using application waypoints. In data-parallel applications, the completion of every single workload task is used as a waypoint *and* as an opportunity to collect intermediate results at the result aggregator (in some cases, the arbitrator itself). Without any loss in generality, let us assume that the tasks in the data-parallel application are homogeneous. Let v_n be the *rate of application waypoints* from a SP n, which is estimated using information obtained during application profiling and during resource allocation. The number of assigned tasks, the time taken for computation δ_n^s, and the time taken for communication δ_s^d at the different SPs are taken into account in this estimation procedure. Therefore, the arbitrator has an initial estimate of the completion time t_n^{end} at n.

As shown in Fig. 2a, every service provider has a deadline $t^0 + \delta_n^{max}$ for the completion of all the tasks that it has been allocated. Here, $\delta_n^{max} < \delta^{max}$ in order to give a margin for result aggregation. As the actual rate of waypoints v'_n from the service provider n will be different from the initial estimate v_n, the arbitrator should be able to absorb this variation and react appropriately depending on where the projected completion time $t_n^{end'}$ of all the allocated tasks lies in relation to the individual service provider deadline t_n^{max}. We identify *four end zones*, namely, Blue, Green, Amber, and Red, as depicted in Fig. 2: the arbitrator's reaction will depend on the zone in which the projected completion time $t_n^{end'}$ falls in. This projection is straightforward as the arbitrator is aware of the number of tasks allocated to each service provider and of the corresponding v estimates.

Red zone (high risk/failure): When $v'_n \ll v_n$ and $t_n^{end'}$ falls in the Red zone (beyond t_n^{max}), it means that the SP n is completing the allocated tasks very slowly or the waypoints are not received at the arbitrator. The arbitrator determines that the particular SP is unable to complete all the tasks that

have been allocated to it within t_n^{max} and reallocates "all" the residual tasks to backup SPs. There is, however, a possibility that this scenario arises due to a very poor estimation of v_n, in which case the profile needs to be updated. This will be done only after ensuring repeatability of this issue.

Amber zone (low risk): When $v'_n < v_n$ and $t_n^{end'}$ falls in the Amber zone (as shown in Fig. 2a), it means that the SP n is completing the allocated tasks slower than what was estimated during profiling. The Amber zone is bounded by t_n^{min} and t_n^{max}. Even though the projected completion time falls within the SP deadline, the uncertainty is deemed unacceptable by the arbitrator, which reallocates a "fraction" of the residual tasks to backup service providers. The arbitrator wants the lower bound of the Amber zone t_n^{min} to be as close as possible to the deadline t_n^{max} so to absorb the effect of acceptable variation in v'_n from v_n. On the other hand, the arbitrator also takes into account the capabilities of the backup resource pool to complete the reallocated tasks within the deadline. Therefore, δ_n^{min}—which determines t_n^{min}—is a function of multiple factors, i.e., $\delta_n^{min} = \Phi(v_n, v'_n, \widetilde{N}, \widetilde{T}, t_n^{end}, t_n^{max})$. When the average size of the resource pool $\widetilde{N}$ and the average availability duration $\widetilde{T}$ of the resource pool are large, δ_n^{min} is large (Amber zone is smaller), i.e., the arbitrator can wait longer before intervening as the resources at its disposal are capable of quickly finishing the incomplete tasks after a reallocation is triggered. On the contrary, when the resources at the disposal of the arbitrator are limited, the Amber zone is larger as the arbitrator and the resource pool need more time to react after a reallocation is triggered.

Green zone (no risk): When $t_n^{end'}$ falls in the Green zone, it means that the current operating conditions at the SP is not significantly different from the one during profiling. This rate of progress is close to the desired trajectory for workload task completion and the arbitrator does not react in such a scenario. In other words, the level of uncertainty is acceptable to still complete all the allocated tasks within t_{max}^n.

Blue zone (no risk but wasteful): When $v'_n > v_n$ and $t_n^{end'}$ falls in the Blue zone, it means that the SP n is completing the allocated tasks faster than what was estimated during profiling. This may happen when the profiling is done under non-ideal conditions. The arbitrator uses this opportunity to tune the application profile for the specific hardware and software platform on the service provider n. In some cases, the arbitrator may deem it undesirable if additional energy (battery drain) is being used up for an unnecessary earlier completion of the task and may reallocate the tasks to other service providers to avoid energy wastage. The upper bound for Blue zone t_n^{min} is a tunable parameter that the application developer can set. In our prototype, we set $t_n^{min} = t_n^{end} - \delta_n^{min}$.

While the dynamics are presented at the SP level in Fig. 2, the same phenomenon can be explained at the arbitrator level by combining waypoint information from all the SPs. Application waypoints could be also seen as indicators of progress similarly to the approach described in [19], which provided a general-purpose API and runtime system to implement progress and performance indicators of individual high-performance computing applications. Similar to the waypoints discussed in this paper, the main purpose of the indicators was improving scheduling policies based on dynamic load-balancing techniques and self-tuning in runtime. The concepts and interfaces proposed in [19] can be extended to implement waypoints also at the mobile-OS layers and not just at the application layer as we do now. Such a general-purpose API, in conjunction with checkpointing [6], will also enable extension of the same principles for self-healing in task-parallel applications with waypoints built "inside" the tasks.

Reallocation strategy: In response to a *partial* or *full* reallocation trigger raised in the Amber or Red zone, the arbitrator provisions additional computing resources and reallocates the workload task(s) so to ensure that the workload is completed within the specified deadline. In contrast to the energy-aware optimization approach for initial task allocation, the reallocation strategy is a heuristic aimed at minimizing the execution time of residual tasks. First, this shift from optimization to heuristic is motivated by the need to react fast. Second, the need to reprovision resources and to reallocate tasks arises due to inaccuracies in the models (for task execution time, service provider availability) used in the optimization approach. Therefore, during recovery, while the models are being tuned for future use, a fast heuristic approach is employed. *We propose a best-fit heuristic to reallocate the incomplete tasks to the backup service providers.*

Backup resource pool: The backup pool is first created and the tasks are allocated to these SPs in pool using the best-fit allocation principle [20]. The eligibility criteria for a SP to be part of the backup pool is $t_n^{start} + \widetilde{T} > t^{max}$. The eligible service providers are then arranged in the decreasing order of i) the rate of application waypoints (primary key) and ii) residual availability duration (secondary key). This ordering gives preference to faster SPs with longer average availability duration.

Task allocation: The fraction of tasks that have to be reallocated depends on the zone in which the projected completion time lies. When $t_n^{end'}$ is in the Red zone, "all" the incomplete tasks at the SP n are reallocated; whereas when $t_n^{end'}$ is in the Amber zone, a "fraction" of tasks at the service provider n are reallocated. This fraction is a tunable parameter that can either be set to a predefined value or be set on the fly based on the capability of the backup pool (size as well as heterogeneity). In our prototype, we set the fraction to be the *nearest quartile of the number of incomplete tasks*.

The fit criteria in the best-fit heuristic is maximization of the minimum residual idle time across SPs after the allocation of all incomplete tasks. The philosophy behind this reallocation approach is load balancing in the backup resource pool so to minimize makespan while at the same time using as few resources as possible. This reallocation does not incur significant computational overhead as the size of the allocation problem is very small. If N is the number of service providers in the backup pool and M is the number of tasks to be reallocated, then the time complexity is given by $\mathcal{O}(M \cdot \log M + M \cdot N)$, where the first component is due to the sorting procedure. Also, note that our heuristic is different from the best-fit-decreasing algorithm for bin packing as we order the bins (SPs) and not the objects (tasks).

TABLE 1
Heterogeneity of Computing Devices in the Testbed

	Samsung Galaxy Tab	Motorola Atrix 2	Samsung Galaxy S	LG Optimus	HTC Desire HD	Dell Netbook	Dell Inspiron Laptop
CPU	1 GHz Dual-core ARM	1 GHz Dual-core ARM	1 GHz ARM	600 MHz ARM	1 GHz ARM	1 GHz Atom	2 GHz Dual-core Intel
Memory (RAM)	1 GB	1 GB	512 MB	512 MB	786 MB	1 GB	2 GB
Network	802.11 b/g/n, Blue tooth	2 G, 3 G, 4 G, 802.11 b/g/n, Bluetooth	2 G, 3 G, 802.11 b/g/ n, Bluetooth	3 G, 802.11 b/g, Bluetooth	2 G, 3 G, 802.11 b/g/n, Bluetooth	802.11 b/g/n	802.11 b/g, Bluetooth
Battery capacity	7,000 mAh	1,740 mAh	1,500 mAh	1,500 mAh	1,400 mAh	4,500 mAh	5,000 mAh
Battery voltage	4 V	3.8 V	3.8 V	3.7 V	3.8 V	11.1 V	11.1 V
Workload completion time	150 s	300 s	390 s	590 s	340 s	100s	35 s

3 Evaluation

We have implemented a small-scale prototype of the proposed autonomic framework and performed an empirical evaluation. We have also used simulations to show the scalability of the proposed framework beyond 10 nodes (the size of our testbed). In the following sections, first, we present details about our testbed and experiment methodology. Then, we discuss specific experiment scenarios and the results that demonstrate the uncertainty-aware self-organization, self-optimization, and self-healing properties of our proposed framework.

3.1 Testbed and Experimental Methodology

Heterogeneous devices: The testbed consists of Android- and Linux-based mobile devices with heterogeneous capabilities (summarized in Table 1). In our prototype, communications among the master and workers as well as among the optimizer and workers happen over a scalable peer-to-peer content-based coordination space [21]. The messages in this coordination space are constructed in the form of tuples (XML strings).

The workload: The mobile application that we used for our experiments is distributed object recognition. In this application, the service requester (which is also the data provider) submits an image of any object that needs to be recognized while also specifying a deadline. The predominant workload in this application is matrix multiplication and the most fundamental workload task is vector multiplication, which is assigned to the different service providers. Distributed object recognition is representative of the wide range of data-parallel applications that our framework can support. Table 1 shows the time taken by the different mobile devices to complete all the workload tasks when operating in isolation. As this profiling was done under controlled conditions, the variation in the task completion times was negligible. For near-real-time performance, the execution time needs to be in the order of tens of seconds and this clearly motivates the need to divide the tasks among SPs in the vicinity for speed up.

Application profiling: As the objective of the optimization problem is maximization of minimum residual battery capacity, the amount of battery drain in service providers as a result of running workload tasks needs to be calculated. However, the usage of actual Watt-hour (Wh) values would result in unfair usage of resources in devices with a higher battery capacity. Hence, in order to deal with the heterogeneity of mobile devices with different battery capacities (shown in Table 1), in our prototype the optimization problem uses the battery capacity *percentage* to make allocation decisions. Using percentage values instead of actual Wh values ensures fairness in usage of the heterogeneous pool of service providers. In order to optimize allocation decisions, it is important to get a good estimate of the instantaneous power drawn in the mobile device while running a workload task. Details of the procedure we followed to obtain the completion time and the power consumption profiles are available in Appendix C of the supplementary material, available online.

3.2 Uncertainty-Aware Self-Organization

Simulation: We performed a simulation to ascertain the gain in terms of reduction in number of workload task migrations that can be achieved through uncertainty-aware self-organization. We performed simulations under different operational scenarios with different SP churn rates. To achieve different churn rates, we progressively decreased the average time a SP is associated with an arbitrator. The four scenarios in Fig. 3a represent a progressive increase in the churn rate of the underlying resource pool (with a corresponding decrease in average duration of availability). The number (15 in total) and combination of SPs in the mobile grid, the number of workload tasks, and the deadlines remain the same for all the four scenarios. We used *percentage of migrated workload tasks* to determine the effectiveness of uncertainty awareness. In order to ensure that the uncertainty awareness capability is not dictated by any particular distribution of SP-availability duration, it was picked at random based on i) Normal distribution (with mean, $\mu = 180, 150, 120, 90$ s and standard deviation, $\sigma = 60$ s) and then on ii) Weibull distribution (with scale, $\lambda = 200, 175, 150, 125$ and shape, $k = 4$). Normal distribution is used for its generality while Weibull distribution is the most popular choice amongst statisticians performing reliability (or survivability) analysis [22]. In order to give statistical relevance to our experiments, we performed multiple trials (by picking availability durations from the aforementioned distributions) until we achieved a very small relative confidence interval (less than 10 percent).

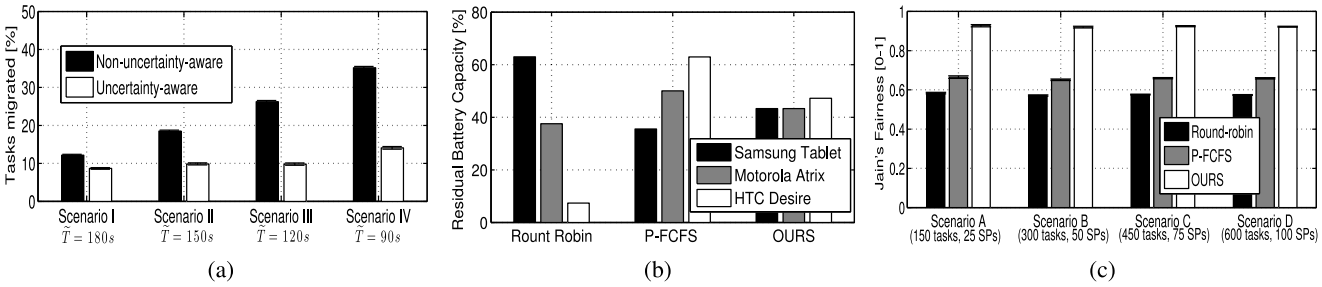


Fig. 3. *Self-organization.* (a) *Simulation:* Percentage of migrated workload tasks when the resource allocation engine is uncertainty-aware and otherwise. *Self-optimization.* (b) *Experiment:* Performance of the proposed framework (in terms of battery drain [percent]) versus P-FCFS and Round-robin approaches; (c) *Simulation:* Performance of the proposed framework (in terms of fairness in battery usage [percent]) across all SPs in comparison with P-FCFS and Round-robin approaches.

Fig. 3a shows how uncertainty awareness at the arbitrator (knowledge of the long-term average of SP availability) helps reduce the number of workload task migrations when the SP-availability duration follows Weibull distribution. Also, another advantage of uncertainty awareness is that it helps decrease churn rate, especially SP departures caused by device users opting out of the application due to undesired battery drain. The SPs will not experience undesired and unfair battery drain, especially the more powerful devices, which are usually preferred for strict deadline requirements. Performance under normally distributed SP-availability duration is available in Appendix D of the supplementary material, available online.

3.3 Self-Optimization

Competing approaches: To assess the self-optimization capability of our framework, we compare it against two competing approaches, i) *Round-robin*, in which the workload tasks are divided equally among all the available service providers and ii) a Pull-based First-Come-First-Served (P-FCFS) [23], in which SPs pull from a bag of tasks at the arbitrator whenever they become idle, work on them, and report the result. Round-robin is chosen for comparison to show the gains (in terms of successful workload completions and battery drain) that can be achieved by exploiting the heterogeneity in computing capabilities of SPs. P-FCFS inherently exploits the heterogeneity in computing capabilities. However, it results in progressively faster devices completing a correspondingly higher number of tasks over time. It is also robust to SP failures or loss in connectivity as it is purely pull based. However, due to lack of self-optimization, there is usually unfair battery drain at the SPs. We performed both experiment and simulation to assess the self-optimization capability.

Experiment: We used three different service providers—a Samsung Galaxy Tab, a Motorola Atrix 2, and a HTC Desire HD—with significantly different computational capabilities and battery capacities (as shown in Table 1). The workload tasks are divided among these service providers based on the result of our resource allocation engine (with a deadline of $100\,\mathrm{s}$) as well as on the two aforementioned competing scheduling mechanisms. The results in Fig. 3b were obtained from 100 consecutive runs of the same workload on the service providers (to achieve a significant battery drain).

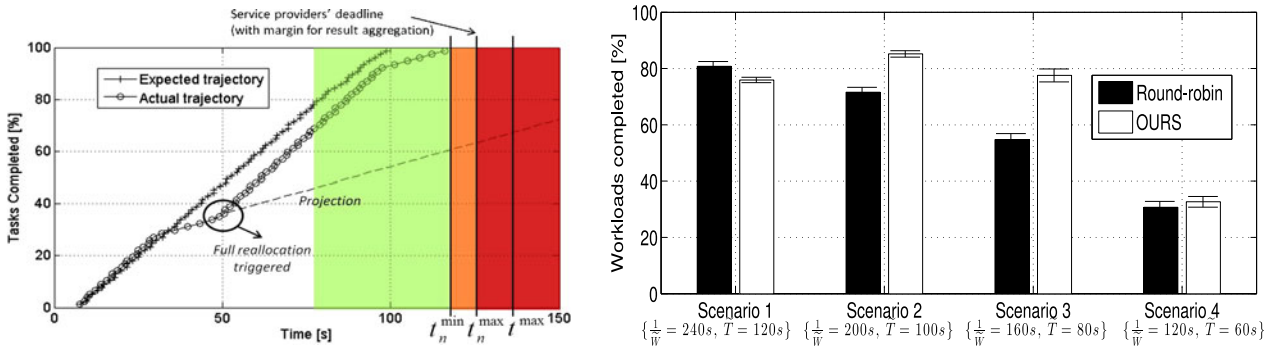
While Round-robin is the slowest of the three as it does not identify and exploit the heterogeneity of the available SPs in terms of their computational capabilities, P-FCFS is the fastest as more tasks are completed by faster devices. Our framework meets the specified deadline by exploiting the heterogeneity of the SPs like P-FCFS. However, the main difference in performance between our solution and P-FCFS as shown in Fig. 3b is that P-FCFS does not appreciate the heterogeneity of devices in terms of battery capacity resulting in asymmetric battery drain.

Simulation: In order to show the scalability of the proposed resource-allocation engine and its performance under difference operational scenarios (in terms of number and combination of SPs), we performed a simulation to ascertain the fairness in battery usage across all SPs when each of the three task-scheduling mechanisms are employed. We used *Jain's fairness index* (1 being the highest and 0 being the lowest) as the measure of fairness. The four scenarios in the table in Fig. 3c represent a progressive increase in the scale and the heterogeneity of the underlying service provider pool as well as the number of tasks. The scaling up is achieved by increasing the resolution of the object's image, which is the input to the object-recognition application. In order to determine the amount of battery drain while using the three task-scheduling mechanisms, we simulated 100 consecutive runs (for significant battery drain) of the workload. This procedure is referred to as one trial. We in turn performed multiple trials, each with a different starting condition in terms of available battery capacities in the service providers.

Fig. 3c shows the average fairness in terms of percentage battery usage at the SPs after each trial. In order to obtain the confidence intervals, we performed multiple trials until we achieved a very small relative confidence interval (less than 10 percent). Our proposed solution achieves the best performance in terms of fairness in the percentage battery usage as it fully exploits the heterogeneity of the devices in the resource pool to successfully complete the workloads within the user-specified deadline.

3.4 Self-Healing

Experiment: We performed an experiment with four SPs—two Samsung Galaxy Tabs, a Motorola Atrix 2, and a HTC Desire HD—to demonstrate the self-healing capability. The requester-specified deadline is $125\,\mathrm{s}$. One of the Samsung Tabs was disassociated from the arbitrator at around $30\,\mathrm{s}$ to show how the arbitrator uses the application waypoints to identify anomalies (such as node failure, disassociation,



(a) (b)

Fig. 4. *Self-healing.* (a) *Experiment:* Demonstration of the use of application waypoints to handle uncertainty (detect node failure) and recover through re-allocation of resources; (b) *Simulation 1.* Effect of high SP churn rate on the percent of successfully completed workloads (Round-robin versus OURS).

etc.) and reacts to it by reallocating incomplete tasks to the available SPs (backup resource pool).

Initially, the workload tasks are divided among three of the four SPs based on the result of our resource allocation engine. Fig. 4a shows the trend of estimated task completion times as seen at the arbitrator over time. At the beginning, the task completion times (actual trajectory) follow the estimated task completion time (estimated trajectory). However, when one of the SPs fails the trajectory deviates and the projected completion time of all the tasks falls in the Red zone. This triggers a *full* reallocation of tasks to the backup pool. During this reallocation, the second Galaxy Tab is used despite its low residual battery capacity as the other two devices alone cannot complete all the tasks within the deadline.

Simulation 1: In order to quantify the effect of increasing churn rates (i.e., high variability in operating conditions) on the number of successfully completed workloads, we performed a simulation the result of which is depicted in Fig. 4b. Churn rate was varied by tuning both the average arrival rate ($\widetilde{W}$) as well as the average availability duration ($\widetilde{T}$) of the SPs. Our proposed framework does not out-perform Round-robin under highly stable as well as under highly varying operating conditions. While comparable performance under stage conditions is self-explanatory, the workload completions under high churn rate is primarily due to self-healing (not due to self-optimization). Note that, however, the proposed framework always out-performs Round-robin in terms of fairness in percentage battery usage across all SPs irrespective of the operating conditions.

Simulation 2: In order to quantify the effect of inaccurate execution-time profiles on the number of successfully completed workloads, we performed a simulation the result of which is depicted in Fig. 5. The mismatch of ground truth with task execution time profiles was simulated by modeling the execution time as a uniform random variable. The lower bound L of the distribution is fixed to 75 percent of the profile while the upper bound U (percent of the profile) is progressively increased as shown in Fig. 5. When the upper bound of the uniform random variable is more than 140 percent of the profile, then the number of successfully completed workload requests (i.e., service requests) starts to lower in comparison to P-FCFS. Note, however, that the proposed solution always out-performs P-FCFS in terms of fairness in percentage battery usage across all SPs irrespective of the operating conditions. In both the simulations, the average number of active SPs was set to 50 and the total number of workloads was set to 100.

4 Conclusions

We proposed a novel resource-provisioning framework for organizing the heterogeneous sensing, computing, and communication capabilities of static and mobile devices in order to a form a mobile computing grid. This local computing grid can be exploited to enable the novel mobile applications that require real-time in-the-field data collection and processing. We imparted the resource provisioning framework with autonomic capabilities, namely, self-organization, self-optimization, and self-healing, in order to be energy and uncertainty aware in the dynamic mobile environment. We performed a thorough performance analysis to verify the autonomic capabilities of the framework via simulations as well as experimental evaluation on a prototype testbed. The response time, quality, and relevance of mobile applications, which rely on real-time in-the-field processing of locally generated data, can be drastically improved using our framework.

Acknowledgments

This work was supported in part by the Office of Naval Research—Young Investigator Program (ONR-YIP) Grant no. 11028418. The authors would like to thank Dr Manish Parashar, Moustafa Abdelbady, and Solomon Lasluisa for their help with the development of the prototype.

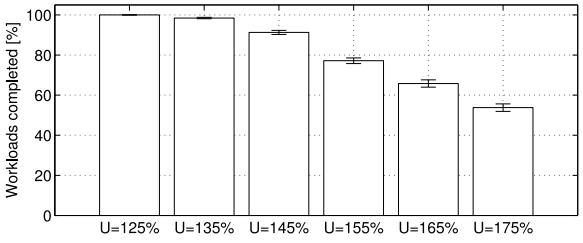


Fig. 5. *Simulation 2.* Effect of inaccurate task execution time profiles on the percent of successfully completed workloads. U is the upper bound of task execution time (percent of the profile) modeled as a uniform random variable.

REFERENCES

[1] D. Anderson and G. Fedak, "The computational and storage potential of volunteer computing," in *Proc. IEEE 6th Int. Symp. Cluster Comput. Grid*, Singapore, May 2006, pp. 73–80.

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

[3] D. Chu and M. Humphrey, "Mobile OGSI.NET: Grid computing on mobile devices," in *Proc. IEEE/ACM 5th Int. Workshop Grid Comput.*, Pittsburgh, PA, USA, Nov. 2004, pp. 182–191.

[4] I. Giurgiu, O. Riva, D. Juric, I. Krivulev, and G. Alonso, "Calling the cloud: Enabling mobile phones as interfaces to cloud applications," in *Proc. ACM/IFIP/USENIX 10th Int. Conf. Middleware*, Urbanna Champaign, IL, USA, Nov. 2009, pp. 83–102.

[5] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "CloneCloud: Elastic execution between mobile device and cloud," in *Proc. 6th Conf. Comput. Syst.*, Salzburg, Austria, Apr. 2011, pp. 301–314.

[6] P. J. Darby and N. F. Tzeng, "Peer-to-peer checkpointing arrangement for mobile grid computing systems," in *Proc. 16th Int. Conf. High-Perform. Distrib. Comput.*, Monterey, CA, USA, Jun. 2007, pp. 211–212.

[7] P. Costa, L. Mottola, A. L. Murphy, and G. P. Picco, "TeenyLIME: Transiently shared tuple space middleware for wireless sensor networks," in *Proc. Int. Workshop Middleware Sens. Netw.*, Melbourne, Australia, Nov. 2006, pp. 43–48.

[8] Y. Huang, S. Mohapatra, and N. Venkatasubramanian, "An energy-efficient middleware for supporting multimedia services in mobile grid environments," in *Proc. Int. Conf. Inf. Technol.: Coding Comput.*, Las Vegas, NV, USA, Apr. 2005, pp. 220–225.

[9] L. dos S. Lima, A. T. A. Gomes, A. Ziviani, M. Endler, L. F. G. Soares, and B. Schulze, "Peer-to-peer resource discovery in mobile grids," in *Proc. 3rd Int. Workshop Middleware Grid Comput.*, Grenoble, France, Nov. 2005, pp. 1–6.

[10] N. Jiang, C. Schmidt, V. Matossian, and M. Parashar, "Enabling applications in sensor-based pervasive environments," in *Proc. BaseNets Conjunction Broadband Commun., Netw., Syst.*, San Jose, CA, USA, Oct. 2004, pp. 871–883.

[11] W.-H. Liao, J.-P. Sheu, and Y.-C. Tseng, "GRID: A fully location-aware routing protocol for mobile ad hoc networks," *Telecommun. Syst.*, vol. 18, nos. 1–3, pp. 37–60, 2001.

[12] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "MAUI: Making smartphones last longer with code offload," in *Proc. 8th Int. Conf. Mobile Syst., Appl., Serv.*, San Francisco, CA, USA, Jun. 2010, pp. 49–62.

[13] M.-R. Ra, A. Sheth, L. Mummert, P. Pillai, D. Wetherall, and R. Govindan, "Odessa: Enabling interactive perception applications on mobile devices," in *Proc. 9th Int. Conf. Mobile Syst., Appl., Serv.*, Bethesda, MD, USA, Jun. 2011, pp. 43–56.

[14] M. Conti and M. Kumar, "Opportunities in opportunistic computing," *IEEE Comput.*, vol. 43, no. 1, pp. 42–50, Jan. 2010.

[15] A. Passarella, M. Kumar, M. Conti, and E. Borgia, "Minimum-delay service provisioning in opportunistic networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 8, pp. 1267–1275, Aug. 2011.

[16] H. Viswanathan, E. K. Lee, and D. Pompili, "An autonomic resource provisioning framework for mobile computing grids," in *Proc. 9th Int. Conf. Auton. Comput.*, San Jose, CA, USA, Sep. 2012, pp. 79–84.

[17] H. Viswanathan, E. K. Lee, and D. Pompili, "Enabling real-time in-situ processing of ubiquitous mobile application workflows," in *Proc. IEEE 10th Int. Conf. Mobile Ad-Hoc Sens. Syst.*, Hangzhou, China, Oct. 2013, pp. 324–332.

[18] J. D. C. Little, "A proof for the queuing formula: $L = \lambda W$," *Oper. Res.*, vol. 9, no. 3, pp. 383–387, May 1961.

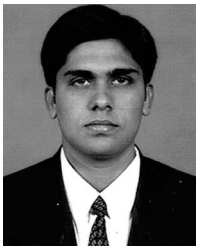
[19] I. Rodero, F. Guim, J. Corbalán, and J. Labarta, "Design and implementation of a general-purpose API of progress and performance indicators," in *Proc. Int. Conf. Parallel Comput.*, Sep. 2007, pp. 501–508.

[20] K. Maruyama, S. K. Chang, and D. T. Tang, "A general packing algorithm for multidimensional resource requirements," *Int. J. Parallel Program.*, vol. 6, pp. 131–149, 1977.

[21] Z. Li and M. Parashar, "Comet: A scalable coordination space for decentralized distributed environments," in *Proc. 2nd Int. Workshop Hot Topics Peer-to-Peer Syst.*, Jul. 2005, pp. 104–111.

[22] G. S. Mudholkar, D. K. Srivastava, and G. D. Kollia, "A generalization of the Weibull distribution with application to the analysis of survival data," *J. Amer. Statist. Assoc.*, vol. 91, no. 436, pp. 1575–1583, Dec. 1996.

[23] H. Kim, Y. el Khamra, I. Rodero, S. Jha, and M. Parashar, "Autonomic management of application workflows on hybrid computing infrastructure," *Telecommun. Syst.*, vol. 19, no. 2/3, pp. 75–89, Feb. 2011.



Hariharasudhan Viswanathan received the BS degree in electrical and computer engineering (ECE) from the PSG College of Technology, India, in 2006, and the MS degree in ECE from Rutgers University in 2009. He is working toward the PhD degree at the Department of Electrical and Computer Engineering (ECE), Rutgers University, since 2009. Currently, he is pursuing research in the fields of mobile computing, datacenter management, and wireless networking under the guidance of Dr Dario Pompili at the US National Science Foundation (NSF) Center for Cloud and Autonomic Computing (CAC). He is a student member of the IEEE.

Eun Kyung Lee received the BS degree in electrical and telecommunications engineering from Soongsil University, South Korea, in 2002, and the MS degree in electrical and computer engineering (ECE) from Rutgers University, in 2009. He is working toward the PhD degree at the Department of ECE, Rutgers University, since 2009. He is currently under the guidance of Dr Dario Pompili as a member of the US National Science Foundation (NSF) Center for Cloud and Autonomic Computing (CAC). His research interests include energy-efficient datacenter management and wireless sensor networks. He is a student member of the IEEE.

Ivan Rodero received the integrated BS and MS degrees in 2004, and the PhD degree from the Technical University of Catalonia, Spain, in February 2009. He is an assistant research professor at Rutgers University. His research interests include the broad area of parallel and distributed computing, and include high-performance computing, energy efficiency, autonomic computing, grid computing, cloud computing, data analytics at extreme scales, and cybersecurity. His current research activities include the key area of green computing with a specific focus on energy efficiency and scalable data management and analytics. He is a member of the IEEE.

Dario Pompili received the 'Laurea' (integrated BS and MS) and the doctorate degrees in telecommunications and system engineering from the University of Rome "La Sapienza," Italy, in 2001 and 2004, respectively, and the PhD degree in electrical and computer engineering (ECE) from the Georgia Institute of Technology in June 2007. He is an associate professor with the Department of ECE at Rutgers University. He is the director of the Cyber Physical Systems laboratory (CPS-Lab), which focuses on research problems in mobile computing, wireless communications and networking, sensor networks, and datacenter management. He received the prestigious US National Science Foundation (NSF) CAREER'11, ONR Young Investigator Program'12, and DARPA Young Faculty'12 Awards. He is a senior member of the IEEE.