Mobile Grid Computing for Data- and Patient-centric Ubiquitous Healthcare

Hariharasudhan Viswanathan, Eun Kyung Lee, and Dario Pompili
NSF Cloud and Autonomic Computing Center
Department of Electrical and Computer Engineering, Rutgers University, New Brunswick, NJ
e-mail: {hari_viswanathan, eunkyung_lee, pompili}@cac.rutgers.edu

***Abstract*—A new context-aware data- and patient-centric paradigm for ubiquitous healthcare is central to deliver personalized healthcare solutions to the elderly and the physically challenged. However, this new paradigm requires real-time in-the-field processing of wirelessly collected vital signs using inherently complex physiological models and analysis of the processed information (derived physiological parameters) under context (e.g., location, ambient conditions, current physical activity) to extract knowledge about the health condition of patients. As the computational capabilities of biomedical sensor nodes are insufficient to run these models, an innovative resource provisioning framework that harnesses the computing capabilities of under-utilized electronic devices in the vicinity (e.g., smart phones, laptops, tablets, DVRs, medical terminals) to form a mobile computing grid is presented. The framework is imparted with *self-optimization* and *self-healing* capabilities for efficiency and robustness under uncertainty, respectively. The proposed mobile grid management framework serves as a key enabling technology for Internet-of-Things- (IoT-) based next-generation ubiquitous healthcare solutions.**

I. Introduction

The rapid growth of non-invasive sensing and low-power wireless communication technologies has enabled continuous monitoring of mobile patients using compact biomedical sensor nodes. These small wearable devices - limited in memory, energy, and computation and communication capabilities - are capable of continuously monitoring vital signs such as blood pressure, temperature, Electrocardiogram (ECG), Electromyogram (EMG), oxygen saturation, and CO_2 concentration. Recently, researchers have developed signal processing algorithms/models that are inherently compute intensive to extract non-measurable physiological parameters (e.g., heart-rate variability, vascular stiffness, peripheral resistance, pulse-transfer time) from these vital signs in order to gain valuable knowledge about the psychophysiological condition.

In addition, *context awareness* – defined as the state of knowledge of external and internal entities that cause a change in the user's situation – is crucial for appropriate interpretation of the vital sign data [1], [2]. Deriving contextual information pertaining to individuals (e.g., physical activity, location) as well as a group of individuals (e.g., degree and pattern of mobility, immediate environmental conditions) requires collaborative in-network processing of heterogeneous sensor data from numerous sources. However, simultaneously executing compute-intensive models for deriving physiological parameters and algorithms for acquiring context awareness in real time requires computing capabilities that go beyond those of an individual sensor node's and/or hand-held device's.

In this paper, we propose to exploit the heterogeneous computing and storage capabilities of static/mobile electronic devices in the vicinity as well as computing clusters in remote datacenters in order to form a *hybrid static/mobile computing grid.* This heterogeneous computing grid can be harnessed to enable innovative *data-* and *compute-intensive* ubiquitous healthcare applications [3] by *collectively* processing massive amounts of vital sign and sensor data in vivo. The electronic devices referred to here are desktop and laptop computers, Digital Video Recorders (DVRs), mobile and static medical terminals, tablets, and smart-phones as shown in Fig. 1.

There are numerous research challenges associated with the realization of our envisioned approach due to the inherent *uncertainty* in the heterogeneous computing infrastructure in terms of network connectivity and device availability. This uncertainty can be attributed to unpredictable node mobility, varying rate of battery drain depending on usage, hardware failures. In order to realize our envisioned approach for ubiquitous healthcare, we make the following significant contributions: 1) a novel energy-aware resource allocation framework for handling resource discovery, service request arrivals, and optimal (in terms of energy consumption) workload task distribution and management; 2) the innovative concept of *application waypoints* to monitor continuously the effect of the aforementioned uncertainties on application performance.

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. As different workloads (e.g., compute-intensive physiological models, algorithms for acquiring context awareness) have different computational, storage, and deadline requirements, waypoints impart the desired robustness and help us eliminate unrealistic assumptions such as accurate knowledge of workload performance on different mobile hardware and software platforms. In this paper, we also present a detailed case study on how a compute-intensive machine-learning-based activity recognition solution (for context awareness) [4] can leverage the proposed framework for uninterrupted operation (ubiquitous) and near-real-time performance.

Our aforementioned contributions overcome the primary impediments to the advancement of in-vivo health monitoring and intervention, namely, i) insufficient computing capabilities on individual sensor nodes and hand-held devices, ii)

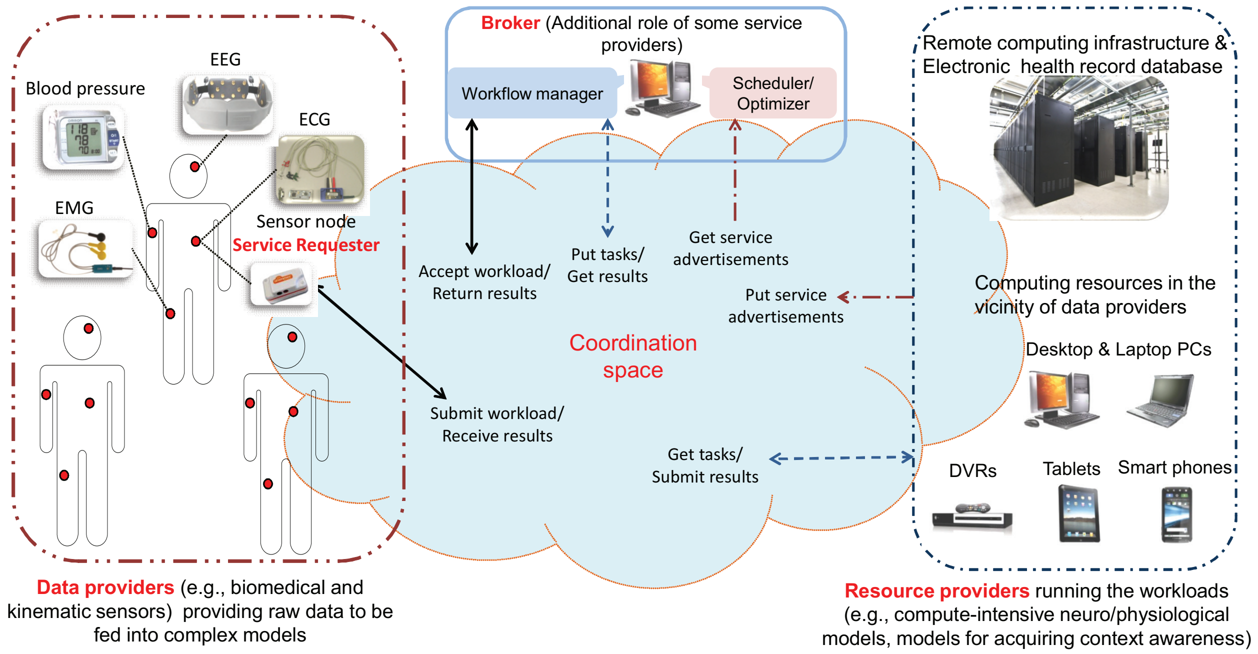

Broker (Additional role of some service providers)
Workflow manager
Scheduler/ Optimizer
Remote computing infrastructure & Electronic health record database
EEG
Blood pressure
ECG
EMG
Sensor node
Service Requester
Accept workload/ Return results
Put tasks/ Get results
Get service advertisements
Put service advertisements
Computing resources in the vicinity of data providers
Desktop & Laptop PCs
Coordination space
Submit workload/ Receive results
Get tasks/ Submit results
DVRs
Tablets
Smart phones
Data providers (e.g., biomedical and kinematic sensors) providing raw data to be fed into complex models
Resource providers running the workloads (e.g., compute-intensive neuro/physiological models, models for acquiring context awareness)

Fig. 1. Concept diagram illustrating our envisioned data- and patient-centric ubiquitous healthcare solution.

prohibitive communication cost and response time involved in enabling real-time psychophysiological analysis using *only* the *wired-grid* and/or *cloud-computing* approaches [5], and iii) the lack of robust mechanisms to ensure application Quality of Service (QoS) under uncertainties. A major prior research effort, mHealth [3], envisions a standardized unified architecture to enable data- and patient-centric ubiquitous monitoring. However, it does not emphasize on real-time processing and analysis of vital-sign data under context as well as uncertainty handling, which are key to enable ubiquitous care. In addition, prior efforts [6]–[10], have aimed at integrating mobile devices into the wired-grid and cloud computing infrastructure mainly as service requesters. In contrast, we exploit mobile devices as service providers and address energy-aware resource management and uncertainty handling for ensuring application QoS even in highly dynamic and unpredictable environments.

The rest of the paper is organized as follows. In Sect. II, we present our resource provisioning framework for mobile grids. In Sect. III, we describe our experimental methodology, results, and a simple case study. Finally, in Sect. IV, we present our conclusions and plans for future work.

II. Proposed Solution

The energy-aware resource provisioning engine will impart self-optimization to our framework while the uncertainty handling mechanism will bestow the self-healing capability.

A. Resource Management Framework

Logical roles: In our solution, the entities of the mobile grid may at any time play one or more of the following three *logical roles* as shown in Fig. 1: 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, resource provider,* or both, and iii) *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 vital sign and contextual 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 broker – an additional role played by some of the service providers – is aided by a novel *energy-aware resource allocation engine,* which will distribute the workload tasks optimally 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 ubiquitous healthcare applications. We advocate the use of *multiple* brokers – each servicing a different subset of data providers simultaneously – in order to avoid a single point of failure and to provide redundancy in case some brokers fail.

We advocate the use of a distributed self-election mechanism for assigning the appropriate number of brokers. Distributed broker self-election is a non-trivial challenge as too many brokers can cause network congestion with excessive control overhead (i.e., communication messages between brokers and data/service providers) while too few can compromise robustness when brokers fail. We leverage our prior experience in distributed adaptive sampling in wireless sensor networks [11] where we were faced with the problem of selective representation to reduce communication overhead while still minimizing the error in reconstruction of the underlying phenomenon. Our self-election mechanism works as follows: each service provider will determine the potential size of its resource pool, i.e., the number of *service advertisements* it has received. Then, all the service providers advertise this number and determine their *rank* in their neighborhood in terms of their *influence* (potential size of their resource pool). The service providers use a pre-determined rank threshold (depending on the network size and density) to elect themselves as brokers.

Service discovery: Service discovery at the brokers is achieved through voluntary service advertisements from the service providers. Service advertisements will include information about the current position, amount of computing (in terms of normalized CPU cycles), memory [Bytes], and communication [bps] resources, the start and end times of the availability of those resources, and the available battery capacity (e_n^{adv} [Wh]) at each service provider n. The broker is aware of the power drawn by the workload tasks of a specific application when running on a specific class of CPU and memory as well as network resources at each service provider as the information about the different types of devices is known in advance.

Workload management: Each broker is composed of two components, namely, *workload manager* and *scheduler/optimizer,* as shown in the top of Fig. 1. 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 tasks among them. The optimizer shares the workload submitted by the data providers among the available service providers based on one of several possible policies. The tasks of a workload may be distributed among the available service providers based on a policy that aims at minimizing the battery drain. Another policy may just place emphasis on response time without considering battery drain. Our framework applies to applications exhibiting *data parallelism* (in which data is distributed across different parallel computing nodes that perform the same task) as well as to applications exhibiting *task parallelism* (in which parallel computing nodes may perform different tasks on different data).

B. Resource Allocation Engine

When a service requester needs additional data or computing resources, it submits a service request to the nearest broker and also specifies δ^{max} [h], the maximum duration for which it is ready to wait for a service response. The resource allocation engine at the broker determines 1) $\mathbf{A} = \{a_{ij}\}_{N\times N}$, the associativity of data provider i with service provider j, 2) $\overline{U} = \{u_n\}_{1\times N}$ (with $u_n \in \{1,0\}$), the list of resource providers to use, 3) $\overline{\Delta}^d = \{\delta_n^d\}_{1\times N}$ [h], the duration for which the services of each service provider will be used for data collection, and 4) $\overline{\Delta}^s = \{\delta_n^s\}_{1\times N}$ [h], the duration for which the resources of each service provider will be used for computation and/or for multi-hop communication as a relay node. The objective of the optimization problem is *maximization of minimum residual battery capacity* at all the service providers while ensuring that the service response is delivered within δ^{max}. This objective maximizes the lifetime of every single service provider and, thus, maintains the heterogeneity of the resource pool for longer periods.

The set of service providers and the duration for which each of their capabilities are availed are determined by considering the trade-offs among the cost (in terms of battery drain) e_n^{data} [Wh] for transferring the data locally from data providers to the resource providers, the computational cost e_n^{comp} [Wh] for availing the computational capabilities of the resource providers for servicing the request and for aggregating and generating the final response. δ_n^d for a service provider n depends on the amount of data it has to transmit (ω_n [Bytes] as a data provider) or aggregate ($\sum_{i=1}^{N} a_{in} \cdot \omega_n$ [Bytes] as a resource provider) and the availed communication capability. δ_n^s for a service provider n depends on the amount of data it has to process and the availed computing capabilities. The constraints to the optimization problem ensure that i) only a resource provider is chosen to perform the computing, ii) the consumer's deadline for service response is met, iii) a service provider is utilized only for the duration for which its services are advertised to be available, and iv) the advertised battery capacity (e_n^{adv}) is not exceeded.

C. Uncertainty Awareness

We identify the different sources of uncertainties and bestow the resource allocation engine with the desired self-healing properties to guarantee application QoS (in terms of response time) even in the highly dynamic computing environment.

Sources of uncertainty: *Inaccuracy in the estimation of task completion times* is one of the sources of uncertainty that affect application QoS. This is especially true when the workload task (e.g., a new model for computing heart-rate variability from ECG) is submitted by the data provider along with the data and whose behavior is not known in advance at the broker. The uncertainty can be reduced to a certain extent in our biomedical application scenario by profiling the behavior of the workloads (physiological models) in advance. However, some models exhibit radically different behaviors depending on different types of inputs (e.g., sorted/unsorted, dense/sparse). *Inaccurate estimation of the availability (duration) of service providers* is another source of uncertainty. The optimization problem may *over or under provision* computing resources due to the aforementioned reasons. Over-provisioning would result in unnecessary wastage of energy (battery drain) while under-provisioning would result in violation of QoS.

Waypoints: We introduce the novel idea of *application waypoints*, at which the service providers report to the broker with their estimate of their *residual task completion time.* Application waypoints could be also seen as indicators of progress and performance. While reporting to the broker, the service providers also specify the next waypoint, i.e., the time at which the broker can expect to hear again from the service provider. If the broker does not receive feedback about the estimated residual task completion time from the service provides at these specified waypoints, it marks these service providers as failed after a timeout and assigns additional resources to take over the incomplete tasks. Also, the frequency of waypoints (depicted as hollow triangles along the x-axes in Fig. 2) is high when the trajectory of the estimated residual task completion time deviates from the optimal trajectory.

In addition to reporting the workload tasks' progress to the broker at the waypoints, the service providers also raise

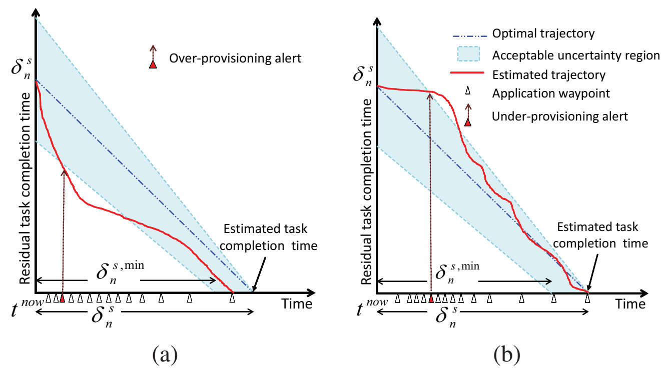


Fig. 2. Illustration of the use of *under-* or *over-provisioning alerts* to convey to the broker that it has (a) under-provisioned or (b) over-provisioned resources and the subsequent correction.

under- or over-provisioning alerts whenever the trajectory of estimated residual task completion time deviates significantly from the optimal trajectory. These alerts are depicted as solid-red triangles in Figs. 2(a) and (b). The aforementioned acceptable degree of deviation is high at the beginning of the execution of a workload task and is low towards the end. These varying degrees of acceptable deviation over time give rise to an *acceptable uncertainty region,* depicted as a blue-shaded cone in Figs. 2(a) and (b). Figure 2(a) depicts an example scenario when under-provisioning of computing resources causes the trajectory of the estimated task completion time to violate the acceptable uncertainty region while Fig. 2(b) depicts an alternative scenario when *over-provisioning* of resources causes the trajectory of the estimated task completion time to violate the acceptable uncertainty region.

Proxies: The brokers play a very important role in handling uncertainties caused by the unavailability of service providers or by the inaccuracy of task-completion-time estimates. In order to ensure that the unavailability of a broker (due to poor connectivity or hardware failure) does not lead to the failure of the entire system, each broker shares with all of its active data and service providers a list of alternate brokers – referred to as *proxies* – ranked according to their proximity (primary key) and physical addresses (secondary key). In case of an broker failure the service providers collaborate with the pre-specified proxy until the end of all active workload tasks. The brokers also share their current state information – namely, data providers and service requests that are currently being served as well as the list of service providers currently employed – with their proxies to handle any unexpected failures. This approach has been exploited previously in grid computing when federating different grids, each with its own resource broker [12], [13].

III. Performance Evaluation

In the following sections, first, we present details about our small-scale prototype and our experiment methodology. Then, we discuss specific experiment scenarios and the results that demonstrate the self-optimization and self-healing properties of our proposed framework.

A. Testbed and Experiment Methodology

Heterogeneous devices: The testbed consists of Android-based mobile devices with heterogeneous capabilities, namely, Samsung Galaxy Tab, Motorola Atrix 2, and HTC Desire HD, with significantly different computational capabilities and battery capacities. In our prototype, communications among the broker and service providers happen over *Comet Space* [14], a scalable peer-to-peer content-based coordination (tuple) space developed at the Cloud and Autonomic Computing Center, Rutgers University.

The workload: The mobile application that we used for validating the self-optimization and self-healing properties of the framework 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-intensive and data-parallel bio-medical applications that our framework can support.* The time taken by the different mobile devices to complete all the workload tasks when operating in isolation is in the order of hundreds of seconds. However, for near-real-time performance, the delay needs to be in the order of tens of seconds and this clearly motivates the need to divide the tasks among service providers 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 experienced by service providers as a result of running workload tasks needs to be calculated. However, the usage of actual Watt-hour (Wh) values will 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 and to ensure fairness, in our prototype, the residual battery capacity percentage is used to make allocation decisions. As the voltage values do not show significant variability, we determined the current drawn in mA, the total time taken for the workload completion, and the time taken to complete one task.

B. 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) *CometCloud* [15], a pull-based task-scheduling mechanism in which the service providers voluntarily pull tasks from the broker, work on them, report the result, and pull the next task to work on.

Round-robin is chosen for comparison to show the gains (in terms of application response time and battery drain) that can be achieved by exploiting the heterogeneity in computing capabilities of service providers. CometCloud inherently exploits the heterogeneity in computing capabilities as it schedules tasks on a First-Come-First-Served (FCFS) basis resulting in progressively faster devices completing a correspondingly higher number of tasks over time. It is also robust to service provider 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 service providers.

Experiment setup: We performed an experiment using three different service providers (a Samsung Galaxy Tab, a Motorola Atrix 2, and a HTC Desire HD) with different computational capabilities and battery capacities. The workload tasks are divided among these service providers based on the result of our resource allocation engine (with a deadline of 100s) as well as on the two aforementioned competing scheduling mechanisms. The results in Fig. 3(a) (workload completion times) were obtained from one run while the results in Fig. 3(b) (residual battery capacity) were obtained from 100 consecutive runs of the same workload on the service providers (to achieve a significant battery drain). While the division of tasks among the service providers remains the same for all 100 runs as far as Round-robin and CometCloud are concerned, the number of tasks to be worked on by the different service providers when our framework is used is determined by the resource allocation engine in every run.

Observations: Figure 3(a) shows the performance of the three approaches in terms of workload completion time. It can be observed that Round-robin misses the deadline and is the slowest of the three as it does not identify and exploit the heterogeneity of the available service providers in terms of their computational capabilities. CometCloud is the fastest as the tasks are distributed on a FCFS basis. Our framework meets the specified deadline by exploiting the heterogeneity of the service providers. The main difference in performance between our solution and CometCloud can be observed in Fig. 3(b), which shows the residual battery capacity after 100 consecutive runs. CometCloud does not exploit the heterogeneity of devices in terms of battery capacity resulting in asymmetric battery drain.

C. Self-healing

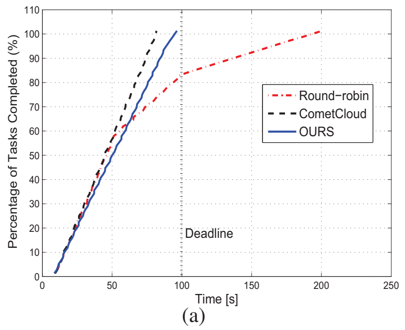
Experiment setup: We performed an experiment with four service providers – two Samsung Galaxy Tabs (with $e^{adv} = 95\%$ and 70%), a Motorola Atrix 2 (with $e^{adv} = 80\%$), and a HTC Desire HD (with $e^{adv} = 90\%$) – to demonstrate the self-healing capability of our resource provisioning framework. The workload tasks were divided among the service providers based on the result of our resource allocation engine (with a deadline of 120s). One of the Samsung Tabs was disassociated from the broker at the time instant 30s to show how the broker uses the application waypoints as well as the service advertisements to identify anomalies (such as node failure, disassociation, etc.) and reacts to it by reallocating incomplete tasks to the available service providers.

Observations: Initially, the workload tasks are divided among three of the four service providers based on the result of our resource allocation engine. The second Galaxy Tab is not chosen initially due to its low residual battery capacity compared to the other devices. Figure 3(c) shows the trend of estimated task completion times as seen at the broker over time. At the beginning of the workload execution, the task completion times follow the estimated task completion time. However, when one of the service provider (a Galaxy Tab) fails, the trajectory of the task completion time violates the acceptable uncertainty region. This violation is detected by the broker in the "detection zone" (the acceptable uncertainty region) and it reallocates the incomplete tasks among the available three service providers so to ensure that the workload is completed within the original estimated time (as seen in the so-called "recovery zone"). 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 specified deadline.

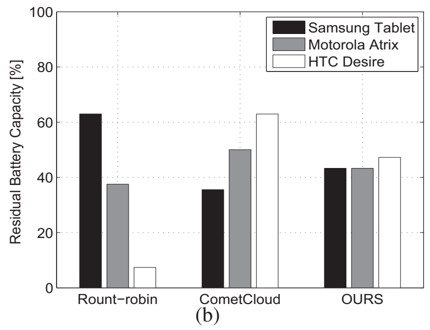
D. Case Study: Real-time Context-aware Health Monitoring

We present a case study to clarify how our mobile grid management framework serves as a key enabling technology for Internet-of-Things- (IoT-) based next-generation ubiquitous healthcare solutions. A new patient-centric paradigm for ubiquitous healthcare characterized by pervasive continuous vital sign data collection, real-time processing of monitored data to derive meaningful physiological parameters, and context-aware data-centric decision making, is central to deliver personalized healthcare solutions to the elderly and the physically challenged. For example, insights into the cardiac health of a subject can be acquired by correlating the Heart Rate Variability (HRV), which can be derived from ECG data, with contextual information such as state of exertion or rest identified using physical activity recognition, which employs real-time distributed sensing and compute-intensive learning-based techniques. In addition to HRV, analysis of the "PQRST pattern" in ECG can provide more insights into the onset of cardiac diseases.

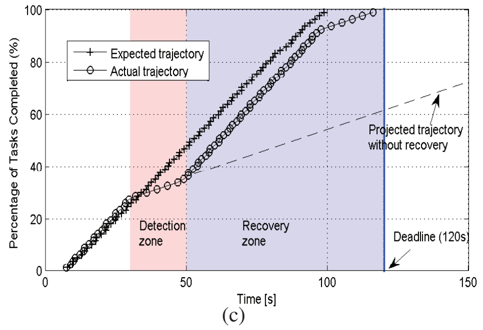
HRV can be determined via frequency- or time-domain analysis of ECG data. We use Shimmer biomedical sensor nodes interfaced with ECG daughter cards for ECG data collection. Similarly, triaxial acceleration as well as angular velocity values (used in physical activity recognition) are obtained from accelerometers and gyroscopes on Shimmer nodes attached to arms and legs. In [4], we proposed a *window-based algorithm* to recognize on the fly various physical activities using kinematic sensor data and a supervised learning approach based on Support Vector Machines (SVMs). We extract meaningful features – such as mean, standard deviation, maximum, peak-to-peak, root-mean-square, and correlation between pair of accelerometer and gyroscope axes – from the raw data and train a SVM to the type of activities that need to be recognized. In the real-time activity recognition phase, which follows this training phase, we work with data collected over a window of duration Δs (typically less than 10 seconds). We extract the aforementioned features from data points within multiple small sliding windows, δ_n (where $\delta_n < \Delta \ \forall n = 1, \ldots, N$), and use the SVM to solve multiple multi-class classification problems so to recognize the activity performed over Δ. The results associated with the different δ_n are ranked based on the confidence associated with each and the activity that corresponds to the highest confidence is chosen as the one performed in the last Δs.



(a)



(b)



(c)

Fig. 3. (a) Performance of proposed framework (in terms of task completion times [s]) versus CometCloud and Round-robin approach. (b) Performance of proposed framework (in terms of battery drain [%]) versus CometCloud and Round-robin approach.(c) Demonstration of the use of application waypoints to handle uncertainty (detect node failure) and recover through re-allocation of resources.

It is clear that real-time processing and analysis of vital-sign as well as kinematic sensor data collected in the same Δ is crucial for in-vivo cardiac health monitoring. The accuracy of both the algorithms (for ECG analysis and activity recognition) increases with increase in Δ. However, their time complexity also simultaneously increases and the computational capability of an individual sensor node (or even a smart phone) is insufficient to produce meaningful results in realistic time bounds for near-real-time performance. In addition, the time complexity and accuracy of the activity recognition algorithm increase simultaneously with the number of small windows used. Our proposed mobile grid computing framework can exploit the *inherent task and data parallelism* in the aforementioned problem scenarios to provide speed up.

Specifically, the two major tasks (ECG analysis and activity recognition) can be offloaded to two different service providers in the vicinity of biomedical sensor nodes as they can be performed independently (task parallelism). The results can be merged at one of the two service providers or a new third one so to analyze the cardiac health under context. Similarly, multiple service providers can be entrusted with the task of activity recognition using different non-overlapping sets of δ_n so as to simultaneously increase the accuracy and speed of computation (data parallelism). A demo of our context-aware ECG monitoring and analysis solution can be found in the Cyber-Physical Systems (CPS) Lab webpage[1].

IV. Conclusions and Future Work

Enabling ubiquitous healthcare applications that require real-time in-the-field vital sign collection and processing using mobile platforms is 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 cloud datacenters for centralized computation. Hence, we proposed 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. We imparted the resource-provisioning framework with self-optimization and self-healing capabilities for energy efficiency and robustness under uncertainties. Currently, we are working on online estimation of the acceptable uncertainty region at the brokers based on the underlying dynamics of the operating environment.

[1]"Demo: Activity Recognition and Vital Sign Monitoring," http://nsfcac.rutgers.edu/CPS/projects/ban/index.php.

References

[1] A. Talaei-Khoei, P. Ray, N. Parameshwaran, and L. Lewis, "A Framework for Awareness Maintenance," *Journal of Network and Computer Applications*, vol. 35, no. 1, pp. 199 – 210, Jan. 2012.

[2] H. Viswanathan, B. Chen, and D. Pompili, "Research Challenges in Computation, Communication, and Context awareness for Ubiquitous Healthcare," *IEEE Comm. Mag.*, vol. 50, no. 5, pp. 92 –99, May 2012.

[3] D. Estrin and I. Sim, "Open mHealth Architecture: An Engine for Health Care Innovation," *Science*, vol. 330, no. 6005, pp. 759–760, Nov. 2010.

[4] J. P. Varkey, D. Pompili, and T. Walls, "Human Motion Recognition Using a Wireless Sensor-based Wearable System," *Personal and Ubiquitous Computing (Springer)*, pp. 1–14, Sep. 2011.

[5] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The Case for VM-Based Cloudlets in Mobile Computing," *IEEE Pervasive Computing*, vol. 8, no. 4, Oct.-Dec. 2009.

[6] D. Chu and M. Humphrey, "Mobile OGSI.NET: Grid Computing on Mobile Devices," in *Proc. of IEEE/ACM Intl. Workshop on Grid Computing*, Nov. 2004.

[7] I. Giurgiu, O. Riva, D. Juric, I. Krivulev, and G. Alonso, "Calling the Cloud: Enabling Mobile Phones as Interfaces to Cloud Applications," in *Proc. of the ACM/IFIP/USENIX Intl. Conf. on Middleware (Middleware)*, Nov. 2009.

[8] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "CloneCloud: Elastic Execution between Mobile Device and Cloud," in *Proc. of The European Professional Society on Comp. Sys. (EuroSys)*, Apr. 2011.

[9] P. J. Darby and N. F. Tzeng, "Peer-to-peer Checkpointing Arrangement for Mobile Grid Computing Systems," in *Proc. of Intl. Conf. on High-Performance Parallel and Distributed Computing (HPDC)*, Jun. 2007.

[10] Y. Huang, S. Mohapatra, and N. Venkatasubramanian, "An Energy-efficient Middleware for Supporting Multimedia Services in Mobile Grid Environments," in *Proc. of Information Technology: Coding and Computing (ITCC)*, Apr. 2005.

[11] E. K. Lee, H. Viswanathan, and D. Pompili, "SILENCE: Distributed Adaptive Sampling for Sensor-based Autonomic Systems," in *Proc. of the Intl. Conf. on Autonomic Computing (ICAC)*, Jun. 2011.

[12] I. Rodero, F. Guim, J. Corbalan, L. Fong, and S. M. Sadjadi, "Grid Broker Selection Strategies Using Aggregated Resource Information," *Future Gener. Comput. Syst.*, vol. 26, no. 1, pp. 72–86, Jan. 2010.

[13] M. Dias de Assunção, R. Buyya, and S. Venugopal, "InterGrid: A Case for Internetworking Islands of Grids," *Concurr. Comput.: Pract. Exper.*, vol. 20, no. 8, pp. 997–1024, Jun. 2008.

[14] Z. Li and M. Parashar, "Comet: A Scalable Coordination Space for Decentralized Distributed Environments," in *Proc. Intl. Workshop on Hot Topics in Peer-to-Peer Systems (HOT-P2P)*, Jul. 2005.

[15] H. Kim, Y. el Khamra, I. Rodero, S. Jha, and M. Parashar, "Autonomic Management of Application Workflows on Hybrid Computing Infrastructure," *Telecomm. Sys.*, vol. 19, no. 2-3, pp. 75–89, Feb. 2011.