Pervasive and Mobile Computing 10 (2014) 3–21

Contents lists available at ScienceDirect

Pervasive and Mobile Computing

journal homepage: www.elsevier.com/locate/pmc

Smartphone sensing offloading for efficiently supporting social sensing applications

Kiran K. Rachuri [a,*], Christos Efstratiou [a], Ilias Leontiadis [a,b], Cecilia Mascolo [a], Peter J. Rentfrow [c]

[a] *Computer Laboratory, University of Cambridge, United Kingdom*
[b] *Telefonica Research, Barcelona, Spain*
[c] *Department of Psychology, University of Cambridge, United Kingdom*

ARTICLE INFO

Article history:
Available online 31 October 2013

Keywords:
Energy efficiency
Phone sensing
Sensing offloading
Smartphones
Social sensing

ABSTRACT

Mobile phones play a pivotal role in supporting ubiquitous and unobtrusive sensing of human activities. However, maintaining a highly accurate record of a user's behavior throughout the day imposes significant energy demands on the phone's battery. In this work, we investigate a new approach that can lead to significant energy savings for mobile applications that require continuous sensing of social activities. This is achieved by opportunistically offloading sensing to sensors embedded in the environment, leveraging sensing that may be available in typical modern buildings (e.g., room occupancy sensors, RFID access control systems).

In this article, we present the design, implementation, and evaluation of *METIS*: an adaptive mobile sensing platform that efficiently supports social sensing applications. The platform implements a novel sensor task distribution scheme that dynamically decides whether to perform sensing on the phone or in the infrastructure, considering the energy consumption, accuracy, and mobility patterns of the user. By comparing the sensing distribution scheme with sensing performed solely on the phone or exclusively on the fixed remote sensors, we show, through benchmarks using real traces, that the opportunistic sensing distribution achieves over 60% and 40% energy savings, respectively. This is confirmed through a real world deployment in an office environment for over a month: we developed a social application over our frameworks, that is able to infer the collaborations and meetings of the users. In this setting the system preserves over 35% more battery life over pure phone sensing.

1. Introduction

The proliferation of smartphones with sensing capabilities has created an opportunity to design systems that capture vast amounts of information about people's social behavior. By leveraging the device's sensing and communication capabilities, applications can capture an accurate depiction of the user's social context, which enables the design of novel applications that can enhance the user experience [1], improve productivity [2], or facilitate new business opportunities such as targeted advertisements [3]. It is expected that the next generation of mobile applications will use continuous sensing of social context at an extremely fine granularity.

* Corresponding author. Tel.: +44 1223763531.
E-mail addresses: kiran.rachuri@cl.cam.ac.uk, kiran.rachuri@yahoo.com (K.K. Rachuri), christos.efstratiou@cl.cam.ac.uk (C. Efstratiou), ilias@tid.es (I. Leontiadis), cecilia.mascolo@cl.cam.ac.uk (C. Mascolo), pjr39@cam.ac.uk (P.J. Rentfrow).

1574-1192/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.pmcj.2013.10.005

A major challenge in achieving accurate social sensing is the significant impact that continuous sensing has on the phone's battery life. The detection of social context through mobile phone sensing, typically leverages a wide range of sensing modalities. Systems such as CenceMe [1], or EmotionSense [4], utilize a combination of accelerometer, Bluetooth, location, and microphone, in order to characterize a particular social situation. The significant energy cost of collecting such information reduces the phone's battery life, and hinders the wider adoption of such applications. Existing efforts to minimize the energy impact rely primarily on adaptive sensing techniques, trading off energy for accuracy with the aim of reducing unnecessary sensor sampling on the device [4,5]. Recently the mobile phone industry made some steps in designing devices that are more suitable for continuous sensing. The latest Apple iPhone 5S,[1] as well as Motorola Moto X[2] feature dedicated chips for continuous activity and audio sensing. The aim of these designs is to reduce the impact of using a power hungry processor to perform sensor sampling, by incorporating a low power micro-controller for such tasks. A multi-processor architecture that divides the computation between a low power processor and the phone's main processor to support a speaker recognition system is presented in [6]. These hardware architectures offer more flexibility for the design of energy efficient continuous sensing applications. However, there is still a significant energy cost for supporting continuous sensing on mobile devices, especially when considering energy demanding sensing activities, such as speech recognition, or location tracking. Although techniques such as adaptive sampling and dedicated context processors have shown improvements in terms of energy consumption, there is still a need for more efficient and complementary solutions that can significantly reduce the cost of continuous sensing before such applications can be widely accepted by everyday users.

In this work, we introduce a novel approach that can offer significantly bigger reductions in energy cost without compromising on the accuracy, by *opportunistically offloading sensing to fixed sensors embedded in the environment*. Most modern buildings are instrumented with a variety of sensors, such as RFID access control systems, Passive Infrared sensors, light sensors, etc. Intuitively, if a mobile application can take advantage of such sensing infrastructures, it could at times suspend local phone sensing, by leveraging remote sensors. For example, relying on a building's access control system, a mobile application can decide to suspend any localization mechanisms on the phone while the user remains in the same building or even room. The feasibility of this approach and the massive energy gains that can be achieved have significant implications both for the design of future mobile applications, and the deployment of sensing infrastructures within smart-buildings. Indeed, such an approach imposes a strong argument for the need of sensing infrastructures that support open access by third party systems, enabling a wide range of mobile and pervasive applications.

In this work, we explore the feasibility of efficient sensing offloading by considering the requirements of social sensing applications operating within a smart-building environment. Through an experimental study within our research institution we demonstrate that the benefits of sensing offloading can only be achieved by considering the potential gain of offloading a particular sensing task versus the potential energy cost incurred primarily due to network communication. Estimating both metrics can depend on a number of parameters and most significantly the behavior of the user and their peers. In this paper, we present the design, implementation, and evaluation of METIS, a sensing platform that implements a novel adaptive sensing distribution scheme that automatically distributes the sensing tasks between the local phone and sensing infrastructure sensors in order to support accurate continuous sensing of social activities. The proposed scheme considers various parameters such as the mobility pattern of the user, duty cycling interval, and cost of sensing to determine whether sensing offload can result in energy gain at any given situation. We show through benchmarks using real traces that the sensing distribution scheme achieves over 60% and 40% energy savings compared to static scenarios where only phone-based sensing is used, and only remote sensors are used, respectively.

Finally, we evaluate the system through a real deployment with 11 users for a month in a working environment using *WorkSense*, a social application that utilizes METIS and aims to raise awareness and improve visibility of social interactions at the workplace, by tracking formal and informal meetings during daily routines, and inferring how social interactions may impact the user's performance. The deployment shows that the application is able to infer the effect of various interaction and social patterns on the work of the users. Furthermore, we show that the system extends battery life by more than 35% compared to when no sensing offloading to the infrastructure is used.

The remainder of this article is organized as follows: In Section 2, we present a study that we conducted to understand the feasibility of the sensing offloading. We then present the design of the METIS system in Section 3 followed by several benchmark tests in Section 4. We present the WorkSense application, and an evaluation of the METIS system and the WorkSense application using a real deployment in Section 5. We present the related work in Section 6, and finally, conclude the paper in Section 7.

2. Saving energy through sensing offloading

The idea of sensing offloading is built on the vision of mobile phone users living in an environment instrumented with a range of sensors that can be accessed over the internet. Within such environment certain pieces of information can potentially be sensed through either the user's mobile device or a sensor that is embedded in the environment. For

1 http://www.apple.com/iphone-5s/specs.
2 http://www.motorola.com/us/FLEXR1-1/moto-x-specifications.html.

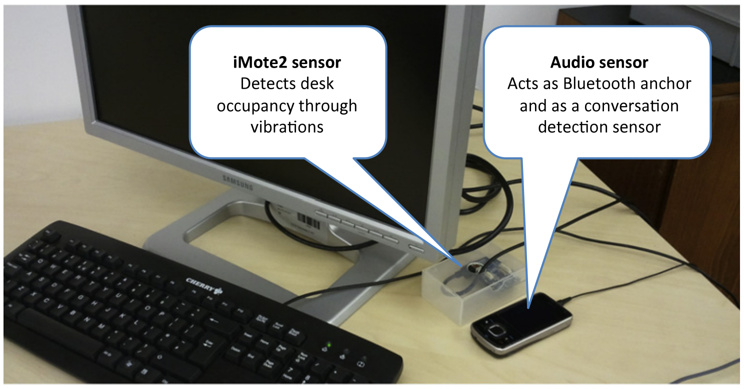
iMote2 sensor
Detects desk occupancy through vibrations
Audio sensor
Acts as Bluetooth anchor and as a conversation detection sensor

Fig. 1. Sensing infrastructure: Each desk is instrumented with a desk occupancy sensor and a Nokia phone acting as conversation detection sensor and Bluetooth anchor point (for enabling indoor localization).

example, detecting if a conversation is taking place in a room, can happen either through the mobile phone's microphone or a microphone in the room, both augmented with the necessary conversation detection software. Within this setting we attempt to explore if sensing offloading can be used to reduce the energy consumption of continuous sensing on mobile phones.

In order to understand the requirements of sensing offloading, and to help us frame our hypothesis, we conducted an *exploratory deployment* within our research institution. The primary objective of the deployment was to help us identify the conditions under which offloading could reduce the energy cost on the mobile device, and those where offloading would lead to higher energy consumption than local sensing. Furthermore, we tried to explore the feasibility of designing a system that can adapt the sensing behavior on the mobile device, selecting the most appropriate action at any given situation.

The focus of the study was to explore the support for social sensing applications. To that end we identified two key sensing modalities: location/co-location sensing, and conversation detection. The experiment was conducted in our research institution involving 10 participants, and 10 offices instrumented with sensors, and lasted one week. During the deployment we collected traces about sensor data both from the mobile devices carried by the participants, and the sensors deployed in the environment. No offloading was used in this study. Using these traces we were able to re-construct the behavior of the system when sensing is offloaded to remote sensors and compare the results with sensing taking place solely on the phone.

2.1. Experimental deployment

We deployed a logging system that was able to collect sensor traces from mobile phones carried by the participants, and sensors deployed in the office environment of our research institution. Both systems were able to detect co-location, location, and conversations of the users.

2.2. Mobile phone sensing

We designed an Android application that was able to perform indoor localization, co-location detection (both using Bluetooth-based localization, utilizing Bluetooth anchors in the environment), and conversation detection (using the microphone). The application was implemented in Java on the Android 2.3.3 platform and gathers data from accelerometer, Bluetooth, and microphone sensors. The conversation recognition module is based on that used in the EmotionSense system [4], which was implemented using the Hidden Markov Model Toolkit (HTK) [7].

2.3. Sensing infrastructure

In our deployment we aimed at exploiting the existing sensing infrastructure in the environment. As part of previous sensing experiments, within our research institution [8], two sensing infrastructures were already available: indoor localization, and room occupancy (Fig. 1). In particular Nokia 6210 Navigator phones were tasked to act as Bluetooth anchors and assist in the localization of mobile phones. The infrastructure includes 12 such Bluetooth anchor points covering a space of 10 offices. Furthermore, each Nokia node periodically scans for Bluetooth devices in proximity using the *lightblue* module for *Python for S60* (PyS60). In order to offer additional support for conversation detection we extended the sensing module on the Nokia devices with conversation sensing functionality (using the same scheme [4]). To capture accurate room occupancy data, a network of imote2 sensors had already been deployed around the office spaces (13 nodes covering 10 offices). The sensors are attached to desks and are able to detect when a particular desk is occupied. The desk occupancy status is inferred by detecting vibration patterns using the 3-axis accelerometer sensors embedded in the node. Each of the nodes periodically sends the current state (i.e., whether the desk is occupied or not) to the root node that is connected to a server. The sensing infrastructure is interfaced with the Internet through an sMap [9] type back-end. sMap exposes a REST-full interface that allows clients to subscribe to sensing events and receive notifications asynchronously.

2.4. Benchmarks

In order to assess the effectiveness of sensing offloading we used the collected traces to investigate the performance using the following schemes.

- *Never offload*. This is the case where no offloading takes place, and the sensing is performed using only the local phone sensors. This scheme resembles the typical behavior of a mobile application where sensing relies solely on the local phone sensors.
- *Always offload.* In this scheme the mobile phone offloads sensing every time it is in an environment where appropriate sensors are available.

The benchmarks consider two sensing modalities: indoor localization using Bluetooth scanning, and conversation detection using microphones. For the always offload scheme, desk occupancy sensors are used to suspend the Bluetooth scanning on the phone when the user is at his desk, and microphone sensors in the environment are used to offload the conversation detection.

In our benchmarks we estimate the energy cost of performing a sensing task locally on the phone, and the cost of communication between the phone and the infrastructure when sensing is performed on the infrastructure. The estimation of local sensing is based on the power consumption values reported by [10,11]. The estimation of network traffic includes the baseline cost of keeping the Wi-Fi interface on and the average cost of communication per byte as it is estimated by [12]. We note that in a realistic setting, users typically enable Wi-Fi for their own purposes, which is further explored in Section 4. The communication traffic between the infrastructure and the mobile device depends on the number of events that are detected by the deployed sensors, and therefore depends on the actual behavior of the participants in the instrumented spaces. Furthermore, in order to understand how these values affect the performance of offloading, we used the same traces to simulate scenarios by modifying certain parameters such as sensing cost or sampling rate. The measurements report the average energy consumption across all the participants for the entire duration of the experiment.

2.5. Lessons learnt

Which sensing tasks to offload

We identified two key parameters in our scenarios that affect the energy cost of sensing: the sampling rate, and the energy cost of sensing and processing data from a particular sensor. In our deployment we had two modalities that can be considered as mid-cost and high-cost respectively in terms of energy. Based on works such as [10,11], a low-cost sensor such as the accelerometer, for example, consumes about 30 mW, the Bluetooth scanning consumes about 160 mW, and an expensive sensor such as GPS consumes around 430 mW. In order to generalize from our traces, we simulated the performance of the two schemes considering a range of sensor sampling costs. Fig. 2, as expected, shows that for low cost sensing, offloading does not lead to energy gain, as the cost of network communication outweighs the small benefit of suspending local sensing.

The varying sampling rate has a similar effect. Higher sampling rate incurs more energy cost on the device, however, it may increase the number of events reported by the infrastructure (fewer missed events). We analyzed the impact of sampling rate by sub-sampling our original dataset. As illustrated in Fig. 3 the increase in sampling interval (decrease in sampling rate) reduces the sampling cost faster than the network cost for offloading. The figure demonstrates the overall cost for Bluetooth scanning with increasing sampling interval, as shown, at low sampling rates, local sensing can in fact perform better than offloading.

The impact of mobility

Apart from the parameters affecting the cost of local sensing, efficient offloading depends on the varying cost of network communication that includes the cost of "hand-over" (subscribe/unsubscribe to a sensor), and the cost of receiving events that are detected by the infrastructure. Both these costs depend on the behavior of the users in the instrumented spaces. We explored the impact of the participants' mobility patterns on the cost of sensing offloading.

In our traces, users spent 64% of their time inside their own office and 21% of their time in other offices (Fig. 4) with sensing capabilities. These were promising results showing that on average 85% of the time users would be in an environment where sensing offloading can be used. At the same time, we discovered that although users spend most of their time in such areas, they frequently visited certain places for short periods of time. In Fig. 5 we observe that users visit other rooms for short, 6-min, intervals (e.g., to have a chat with someone or to have a short meeting about work). For those times, where offloading opportunities were available, we analyzed the average time that people spent when they visited a particular location. Fig. 6 shows a CDF of the time each user spent in each visit. We observe a high number of short visits (50% of them last less than 20 min), which can be the cause of increased network traffic due to hand-overs, moreover, there is a possibility of sensing events to be either missed or wrong (reported by the wrong environment).

We investigated the impact of user mobility looking at scenarios with different average times that people spent in a given location. We used the original traces collected and modified the amount of time that each user spent in each room.

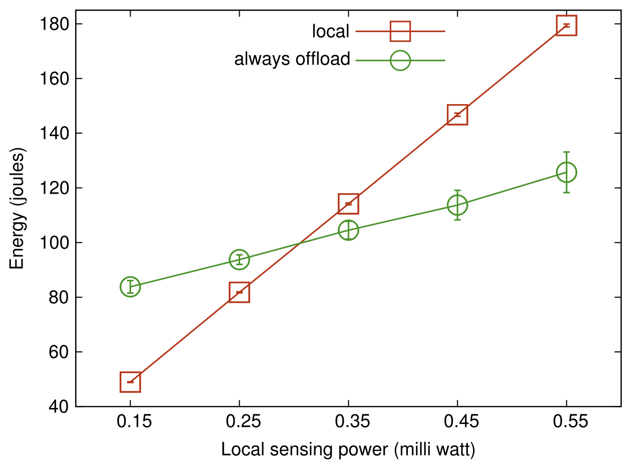

Fig. 2. Energy consumption per hour vs. local phone sensing cost.

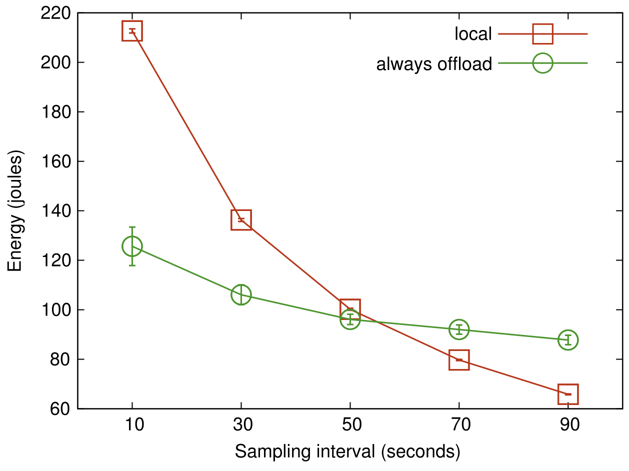

Fig. 3. Energy per hour for location detection vs. sampling interval.

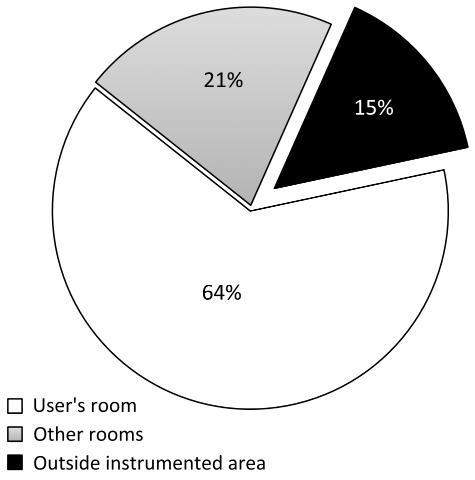

Fig. 4. % of time spent by users in various location types (user's/other rooms were instrumented).

As illustrated in Fig. 7, the results demonstrate how high mobility can significantly increase the cost of sensing offloading. Essentially, for a given visit to a location, offloading can only deliver positive results if an offloading scheme can predict the possible time that a user will spend in that location, and ensure that the duration is enough to offer an energy gain.

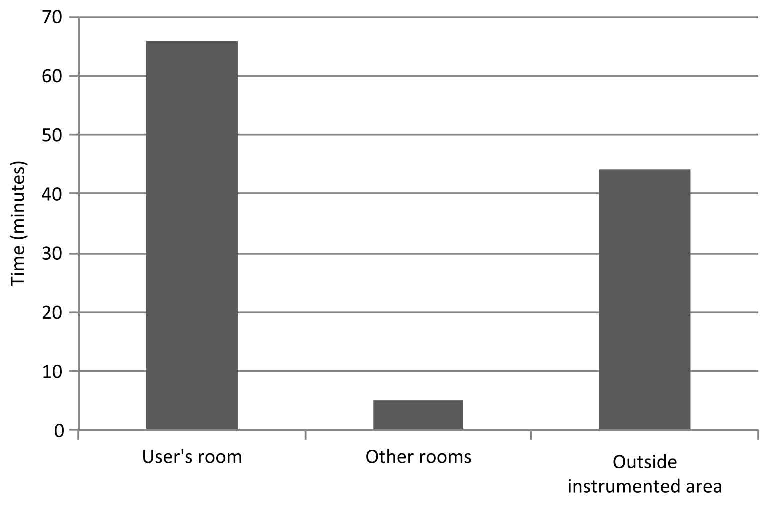

Fig. 5. Average time spent for each visit to a location (user's/other rooms were instrumented).

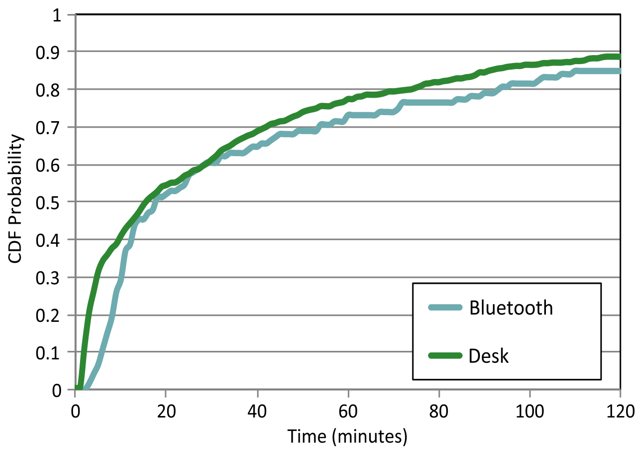

Fig. 6. Cumulative Distribution Function (CDF) of time spent for each location (room) visit.

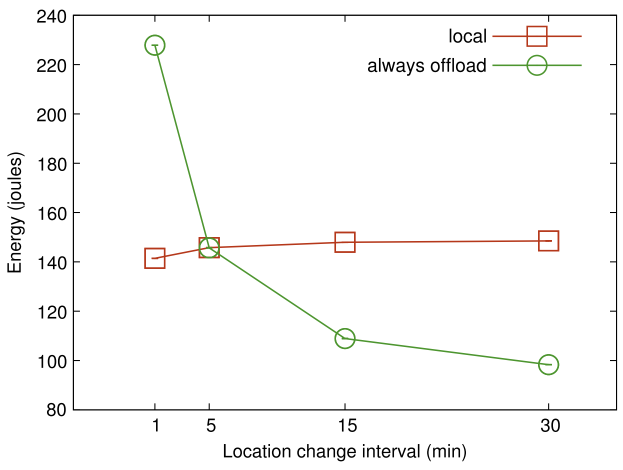

Fig. 7. Energy per hour for location detection with varying mobility patterns.

The results clearly suggest that neither pure phone sensing nor exclusive remote sensing is an optimal solution for all the scenarios, and a dynamic sensing offloading scheme that considers the sensing parameters and the user's mobility is required for achieving an optimal performance.

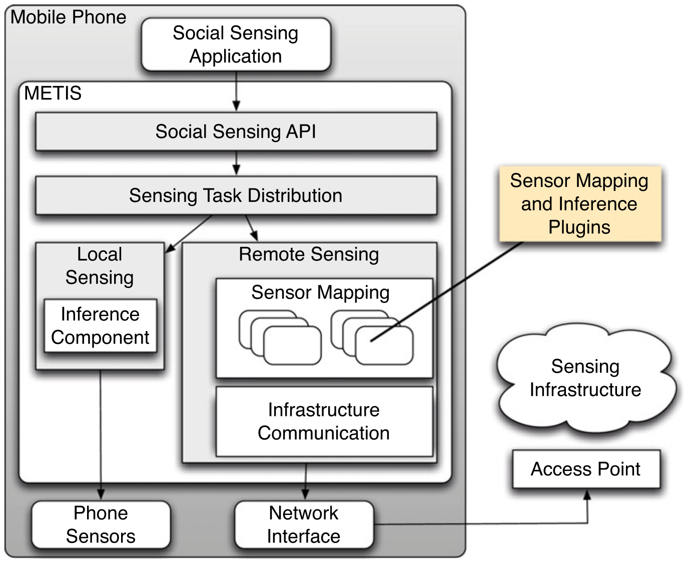

Fig. 8. Architecture of the METIS system.

3. The METIS system

Motivated by the results of the exploratory deployment, we designed the METIS system. METIS is a mobile phone service that offers efficient continuous sensing for mobile social applications by leveraging both phone and infrastructure sensing. It provides an abstraction over the sensing modalities required by typical social sensing applications, such as location, co-location, and conversation detection and offers a framework for the incorporation of additional sensing modalities when needed (Fig. 8). The operation of METIS includes the discovery of sensing devices that are available in the immediate environment of the mobile phone user, the identification of devices that could be used for offloading, and the decision to perform such offloading in order to maintain overall energy efficiency. If such offloading is not considered beneficial METIS falls back to local sensing utilizing the resources of the mobile device. The primary challenges and the main focus of this work are related to the utilization of remote sensors in order to improve sensing performance, which we address in the following subsections.

3.1. Interaction with sensing infrastructure

One of the key motivations for the design of METIS is the emergence of a range of Web-based architectures that allow interaction with the sensing infrastructure over IP networks. We decided to assume the presence of sensing infrastructures that follow the same architectural principles that are adopted by such architectures. Systems such as SenseWeb [13] identify two key elements in their architecture: the presence of a rendezvous point, which allows a client to query the infrastructure about available sensing resources and their capabilities, and the support for a resource communication protocol that enables clients to interact with specific sensing resources. The operation of METIS imposes the following two requirements on the sensing infrastructure: (i) the specification of the physical location of a sensing resource as reported by the rendezvous point, and (ii) the support for an asynchronous publish–subscribe interface for communication with a sensing resource. Both of these requirements are supported by most common Web-based sensing architectures. In the design of METIS we adopt the sMAP [9] communication protocol for interaction with specific sensing resources.

The information that can be retrieved through the rendezvous point plays a key role in the operation of METIS. The expectation is that METIS can identify the physical location of sensor points. In the design of METIS we target indoor environments with the aim of taking advantage of common sensing technologies that can be found in smart homes or office buildings. To that end, we define a minimal XML schema of the sensing infrastructure that incorporates information about the physical location of sensor points within a building (Fig. 9). Although the format of the schema is designed to meet our needs, the same information can be easily extracted by standard-based schemata such as *SensorML*. METIS can be trivially extended with additional schema parsers to support multiple infrastructure interfaces.

Such information is crucial in order for METIS to identify which types of sensors are available around the user at any given time. Furthermore, an important requirement for the accurate operation of METIS is that a given environment should offer an indoor localization technology at least at room-level accuracy that would enable the mobile device to discover where it is currently within a building. In a typical scenario, when a user enters a building, METIS will attempt to retrieve the sensing infrastructure model from a well-known repository. If the building offers a known indoor localization technology (i.e., Wi-Fi fingerprinting, Bluetooth RSSI trilateration), METIS will utilize this technology to discover the location of the user

```
<?xml version="1.0" encoding="UTF-8"?>
<mt:METIS xmlns:mt="http://our.domain.org/metisML">
 <!-- ........... -->
 <mt:SensorNetwork>
  <mt:sMAPURI>http://our.domain.org/</mt:sMAPURI>
  <mt:SensorList>
   <mt:SensorNode>
    <mt:SensorId>BT_D11_R01</mt:SensorId>
    <mt:SensorType>BluetoothScanner</mt:SensorType>
    <mt:Location>
     <mt:Label type="office">R01</mt:Label>
    </mt:Location>
   </mt:SensorNode>
   <mt:SensorNode>
    <mt:SensorId>ACC_D1_R01</mt:SensorId>
    <mt:SensorType>DeskUseDetector</mt:SensorType>
    <!--.........-->
   </mt:SensorNode>
  </mt:SensorList>
 </mt:SensorNetwork>
</mt:METIS>
```

Fig. 9. Sample sensing infrastructure manifest obtained by the METIS system from a service provider.

in the building. That information is then used to discover the types of sensors that can be accessed by the platform at any given time. We note that there have been many works on Wi-Fi based indoor localization such as RADAR [14], and it has been shown in [15] that accurate room-level localization is possible based on the signal strength of Wi-Fi access points. Although Bluetooth devices are not commonly deployed, Wi-Fi access points are widely deployed, which could be exploited for supporting indoor localization.

Sensor mapping

One of the primary functions of METIS is the association of social sensing modalities with possible remote sensors that can be used for offloading. METIS enables this association with the incorporation of sensor-specific drivers in the form of *plug-ins*. The *Sensor Mapping* component acts as a repository of sensing plug-ins, triggering them on demand when a particular sensor device is within the proximity of the user. Each plug-in maps a specific high-level social sensing task to a combination of subscriptions to certain sensor nodes in the environment. The design of such plug-ins is non-intuitive and depends on the presence of sensors with specific features in the environment. As the main aim of METIS is to reduce the energy cost on the local device, such mappings aim at reducing the frequency of local sensing by relying on notifications that can be received from the environment. An inference object incorporated in the plug-in is responsible for extracting meaningful social events from the event notifications received from the infrastructure. The receipt of events from such nodes can allow the plug-in to derive inferences and report information about the location of the user, or the current activity state, for example. A sensor task should only be offloaded to a suitable remote sensor, for example, offloading a speaker recognition task to a remote microphone may not always provide the same level of accuracy as the phone's microphone, therefore, in this case, the remote sensor can be used as a context change detector from silence to non-silence so that the phone's microphone can be activated on detecting that a conversation has just been started.
Plug-ins that have been implemented for METIS include:

- If real-time room occupancy information is available, subscribe to receive events about the current room, and switch off the location scanning when the number of people in that room has not changed.
- If desk occupancy information is available, subscribe to receive notifications about the current desk, and report current activity as "sitting" without using the accelerometer.
- If noise level detection is available in the room, subscribe to receive notifications about changing noise levels, and adjust conversation detection on the phone when not needed.

3.2. Sensing offloading

By analyzing the results of the exploratory study we were in a position to identify the parameters that can affect the energy trade-offs when deciding to perform sensing offloading to remote sensors. Specifically, the decision is based on the estimation of the energy cost when sensing is performed on the phone, and the prediction of the network energy cost when sensing is performed remotely. In estimating the latter, the user's mobility pattern, the time spent in a particular location, and the rate at which events are reported by the remote sensor are factors that affect the energy cost. In this subsection we describe an offloading scheme that achieves energy efficiency by considering all these parameters.

3.2.1. Gain threshold based offloading

The *Gain Threshold* scheme operates by calculating the probability that offloading a particular sensing task would result in energy gain when compared to the corresponding local sensing task. If the probability of gain is greater than 0.5, i.e., if offloading has more than 50% chance of resulting in gain then the offloading is performed. The probability of gain is calculated by estimating the possible communication costs that the phone may incur if offloading is performed. When a sensing task is offloaded, there are two types of energy costs involved, *fixed* and *variable* costs. The fixed costs are the costs involved in maintaining a network connection between the phone and the infrastructure system, subscribing to a remote service for offloading, and canceling the subscription at the end. The variable costs are the communication costs incurred by updates received as part of the infrastructure sensor events that depend on the user behavior.

A decision to offload a sensing task results in energy gain if the user stays in the location for long enough so that the sum of the fixed and variable costs is less than the cost of local phone sensing for that period. We refer to this minimum time period as *gainTimeThreshold*. We note that fixed costs can be calculated based on offline estimated values for data transfer over the network (for example, this could be measured on the Android phones using a power meter [11] and on the Nokia phones using the Nokia Energy Profiler), and variable costs (or event rate or sensor state change rate) can be calculated based on the past history of event traces as recorded by the sensing platform. The *gainTimeThreshold* varies for each of the sensors as the cost and event rate for these change.

Gain time threshold.

In this subsection, we present the calculation of the *gainTimeThreshold*, and in the next subsection we present the estimation of the probability of achieving gain.
Notation used:

G_s :	gain threshold time of a sensor s
C_{sl} :	cost per sample of local sensing
S_{sl} :	sampling rate of the sensor s
C_{so} :	fixed cost of offloading the sensing task
C_{sr} :	cost per update of remote sensing
C_{nr} :	baseline cost for maintaining network connection
U_{sr} :	update rate of remote sensing
N :	number of sensing tasks that can be offloaded.

According to the definition of *gainTimeThreshold*, for a sensor s, the energy consumption of local phone sensing is equal to the sum of fixed and variable costs of remote sensing for *gainTimeThreshold* amount of time. In other words, if remote sensing is used for more than *gainTimeThreshold* amount of time, then it results in a positive energy gain, and if remote sensing is used for less than *gainTimeThreshold* amount of time, then it results in a negative energy gain, i.e., offloading is not beneficial in this case.

The local sensing cost for G_s amount of time for a sensor s (L_s) = The cost of local sensing per sample (C_{sl}) × Local sampling rate (S_{sl}) × G_s.

The remote sensing cost for G_s amount of time (R_s) = Fixed control traffic cost (C_{so}) + (Event update rate (U_{sr}) × Cost of network transfer per update (C_{sr}) × G_s) + Baseline network connection cost per sensor for G_s amount of time.

Per the definition, G_s is the amount of time for which, local cost = remote cost, i.e., $L_s = R_s$.

$$\Longrightarrow C_{\mathrm{sl}} \times S_{\mathrm{sl}} \times G_s = C_{\mathrm{so}} + C_{\mathrm{sr}} \times U_{\mathrm{sr}} \times G_s + \frac{C_{\mathrm{nr}}}{N} \times G_s \tag{1}$$

$$\Longrightarrow G_s = \frac{C_{\mathrm{so}}}{C_{\mathrm{sl}} \times S_{\mathrm{sl}} - C_{\mathrm{sr}} \times U_{\mathrm{sr}} - \frac{C_{\mathrm{nr}}}{N}}. \tag{2}$$

G_s for a sensor s quantifies the minimum amount of time the sensing task should suspend local sensing and use remote sensing to achieve energy cost benefit i.e., it is the minimum amount of time the user should stay in the current location after offloading the task to achieve energy gain.

Probability of gain estimation

In this subsection we present the estimation of the probability that offloading a sensing task (s) will result in gain. This value is used to make a decision on whether this offloading is beneficial. Let $\{v_{j_1}, v_{j_2}, \ldots, v_{j_k}\}$ be the total visits of the user to a location j, let $\{tv_{j_1}, tv_{j_2}, \ldots, tv_{j_k}\}$ be the total duration of each of the k visits, respectively. First, for each sensor s, we divide the total duration of lth visit to a room j into two parts: *favorable time* (ft_{sj_l}) and *unfavorable time* (ut_{sj_l}). Favorable time for a sensor is the time during which the offloading of the sensing task results in a positive gain, and unfavorable time for a sensor is the time during which the offloading results in a negative gain. Therefore, for a visit to a room, the favorable time is the total visit time subtracted by the G_s value (as we need at least G_s amount of time to achieve positive gain), and unfavorable time is G_s, i.e., for lth visit to a room j by the user, favorable and unfavorable times for a sensor s are calculated as:

$$ft_{sj_l} = tv_{j_l} - G_s \tag{3}$$

$$ut_{sj_l} = G_s. \tag{4}$$

Then, the probability (pt_{sj}) that a user stays at a location j for more than the *gainTimeThreshold* (G_s) value for a sensor s is calculated as: the total favorable time at location j for all visits divided by the total time at the location j.

$$\Longrightarrow pt_{sj} = \frac{\sum_{l=1}^{k} ft_{sj_l}}{\sum_{i=1}^{k}(ft_{sj_i} + ut_{sj_i})}. \tag{5}$$

pt_{sj} is the probability of gain that is used to make an offloading decision for a sensing task s when the user is in room j. Finally, if this probability value is greater than 0.5, it indicates a more than 50% chance of resulting in positive gain, and in this case the sensing task is offloaded. A task that is offloaded to a remote sensor is canceled (unsubscribed from remote sensing updates) when the user moves away from the current location, as the remote sensor may not capture the user's activities accurately, as it is not in proximity to the user. We present a detailed evaluation of this scheme through micro-benchmarks in the next section.

3.2.2. Extensions of gain threshold offloading scheme

Behavioral patterns of users tend to be reasonably regular, which can be exploited to further improve the model just presented. In this subsection, we present an extension of the gain threshold based offloading scheme considering significant dimensions affecting the behavior of users. The objective of these schemes is to improve the estimation of the probability of gain by improving the accuracy of estimating the time a user spends in a room and the estimation of the number of events that could be generated by the remote sensor. In particular, additional parameters considered are time of day, day of week, and identities of co-located users. Users at the workplace and in similar environments have a specific schedule for each time slot of the day, for example, they may stay at the common room for lunch everyday at 12.30 pm for 30 min, however, they might only visit the common room for shorter periods and less regularly at other times of the day. User behavior is also driven by co-located users, for example, when the manager of a group is around, group members may also be around and may spend most of their time in their offices. The main idea is to exploit this behavior in addition to the threshold based offloading.

In these extensions we introduce additional constraints to the derivation of the probability of gain value. Let t, d, and c denote the time of day, day of the week, and the set of co-located users, respectively. The probability of gain value to decide on offloading a sensing task s when the user is in room j is then redefined as:

$$p_{sj} = pt_{sj,t,d,c} \tag{6}$$

where $pt_{sj,t,d,c}$ is the probability that the user stays in the location j for more than *gainThreshold* amount of time when he/she is co-located with the users in the set: c, the current time of the day: t, and current day of the week: d. The favorable and unfavorable time values in deriving $pt_{sj,t,d,c}$ should be calculated according to the constraints t, d, and c.

We could also derive more variations by only considering a subset of these dimensions. For example, if we consider only co-located users, the probability of gain can be defined as:

$$p_{sj} = pt_{sj,c}. \tag{7}$$

Similarly, other variants can be derived i.e., these constraints can be applied in combination, which results in a total of 7 possible ways of estimating the probability of gain value.

4. Benchmarks

In this section we present the evaluation of the gain threshold offloading scheme through micro-benchmark tests. We compared the energy performance of the scheme with the two schemes described in Section 2. The dataset collected from the initial study was used for these tests. In this evaluation, the gain threshold technique works in an online fashion, i.e., as the traces are replayed the gain threshold (Eq. (2)) and the probability values (Eq. (5)) are continually learned and the decision on offloading is taken accordingly.

4.1. Sensing cost, sampling interval, and mobility

We showed in the results of the initial study that the two simple sensing schemes may not be suitable for all conditions. We now compare the performance of the proposed scheme considering varying cost of sensing, sampling interval, and mobility patterns. Figs. 10–12 show the total energy consumption of the schemes with respect to these variables. We can observe that the threshold scheme tends to match the best performing scheme in all the cases as it considers these parameters in its decision to offload.

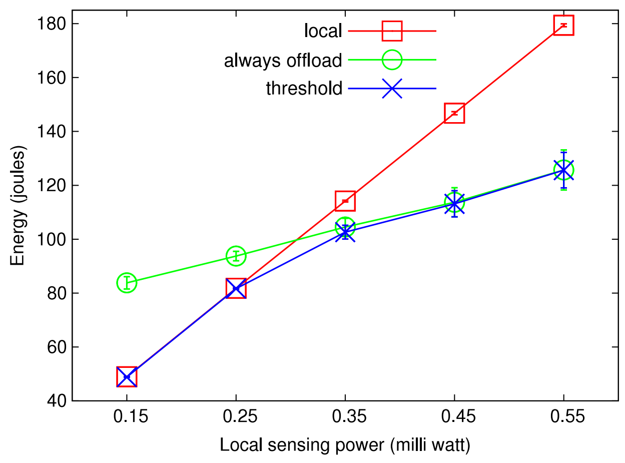

Fig. 10. Total energy consumption per hour for varying sensing cost.

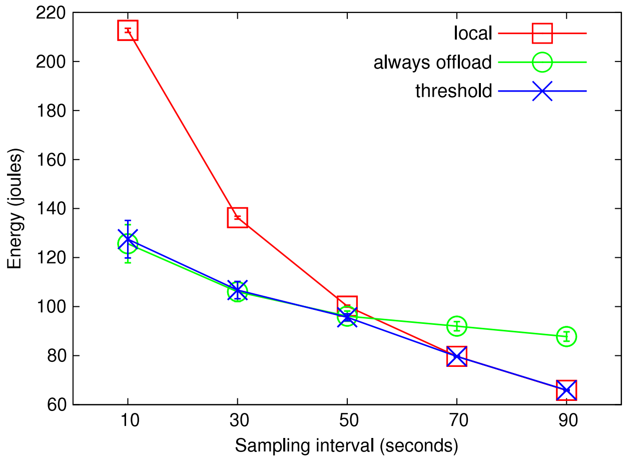

Fig. 11. Total energy consumption per hour for location detection with varying sampling interval.

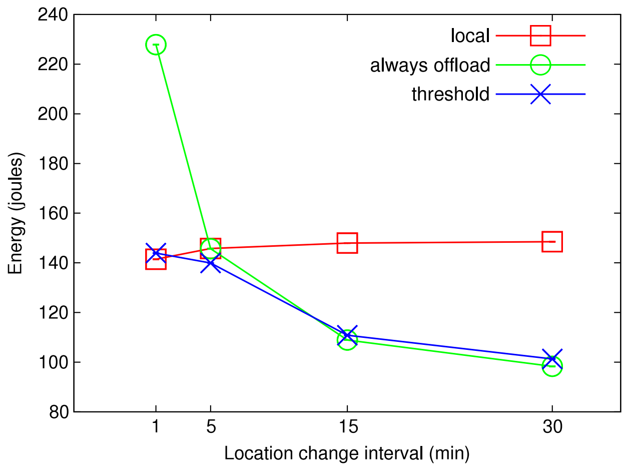

Fig. 12. Total energy consumption per hour for location detection for varying mobility patterns.

4.2. Excluding Wi-Fi baseline cost

In the results presented so far, we include in the network cost function the Wi-Fi baseline cost (cost of keeping Wi-Fi on). However, it is typical for many users to keep Wi-Fi on for other unrelated purposes. In order to analyze this situation,

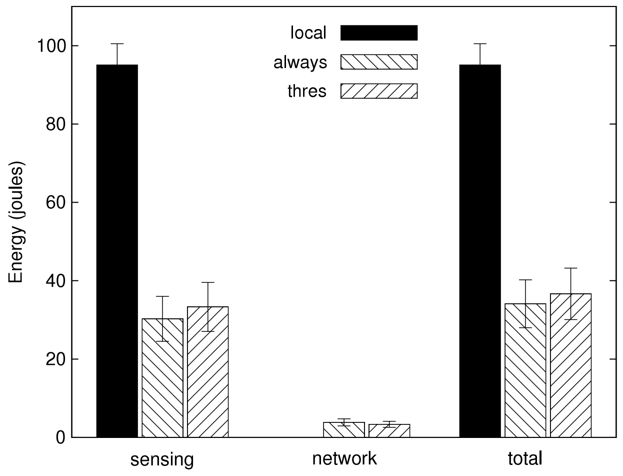

Fig. 13. Energy per hour for location, conversation detection without considering Wi-Fi baseline cost.

we benchmarked the behavior of the scheme excluding the Wi-Fi baseline cost from the calculation considering only the cost of data exchange. Fig. 13 shows the local sensing, network exchange, and total energy consumption for location and conversation detection. We now observe that the offloading schemes result in considerable energy savings due to the fact that the Wi-Fi was already enabled. The threshold scheme resembles the remote sensing scheme in this case, achieving high efficiency, saving around 60% of energy when compared to pure phone sensing scheme.

As shown in the results so far, the threshold scheme tends to follow the most optimum scheme in different circumstances. However, the optimal strategy may not always include one of the extreme offloading schemes. To demonstrate this, we evaluate the schemes with respect to the following two scenarios.

4.3. Adaptive sampling

The use of adaptive sampling is a typical approach in mobile applications that use continuous sensing [4]. In these cases the local phone sensor sampling rate changes in the face of changing behavior of the user. In this test, we used two adaptive sampling schemes presented in [16]: In *adaptive scheme* 1, we used an exponential back-off and a linear advance of sampling interval, and in *adaptive scheme* 2, we used a linear back-off and an exponential advance of sampling interval. The sampling interval is increased (sampling rate is decreased) using a back-off function when there is no change to the user's context and the sampling interval is decreased (sampling rate is increased) using an advance function when there is a change to the user's context. The context of the user for this evaluation is defined as the co-located Bluetooth devices. Therefore, the phone's Bluetooth sensor is sampled at different rates based on the user's current and previous context. If a decision on where to sense (locally on the phone or remotely in the sensing infrastructure) is taken statically then it would not always be optimal as the sampling rate changes with the user's context.

Fig. 14 shows the result of this evaluation. We observe that, in our deployment, for the purely local sensing the exponential back-off (scheme 1) saves more energy than the linear back-off (scheme 2) as the former increases the sampling interval (decreases sampling rate) more aggressively than the latter. Also, the choice of adaptive scheme does not have a significant impact on the *always offload* scheme as it uses remote sensing most of the time (except in uninstrumented areas). However, in both cases neither the *local* or *always* schemes is optimal. This is because when using adaptive sampling, each of them will be optimal for a subset of the possible situations. The threshold scheme appears to outperform the two others, by selecting the most optimal approach in every case as it considers the sampling rate in making offloading decisions.

4.4. Multiple sensors

In this case, we evaluate the schemes using an inexpensive (30 mW) and an expensive (1.2 W) sensors. We assume that the inexpensive sensor generates large amount of data per sensor sampling (100 kB per sample, similar to the audio recording for 5 s using the PCM format in the conversation detection module). Fig. 15 shows the result of this evaluation, where we can observe that the energy consumption of the threshold scheme is much lower than the other schemes, in particular, when the Wi-Fi baseline cost is considered, the threshold schemes consumes 46%, 31% less energy than the always offload and always local schemes, respectively. When the Wi-Fi baseline cost is not considered, the threshold scheme consumes 57%, 63% less energy than the always offload and always local schemes, respectively. The always local scheme consumes a lot of energy because of local sensing of the expensive sensor and the always offload scheme consumes a lot of energy because of the communication cost of the inexpensive sensor. However, the energy consumption of the gain threshold based offloading scheme is much lower than the other schemes, as it selects the most optimal configuration for each sensing modality.

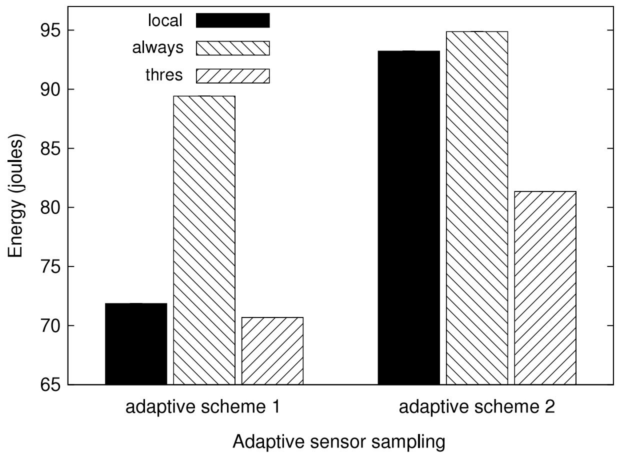

Fig. 14. Energy consumption per hour for two adaptive sampling schemes.

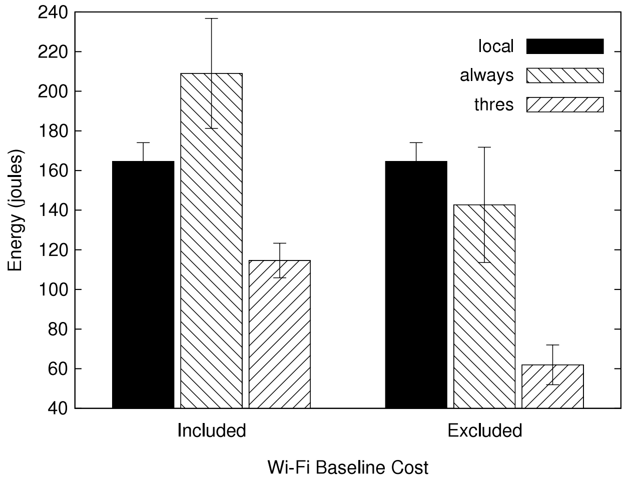

Fig. 15. Energy consumption per hour for an inexpensive and an expensive sensors.

The results presented clearly show that the proposed scheme dynamically offloads the sensing tasks by adapting to the sensing parameters and the user's mobility patterns.

5. Case study

The ultimate evaluation of METIS was performed through a real deployment. The main goals of the deployment were to evaluate the energy efficiency of METIS, and to show that the applications can capture the behavioral patterns of the users utilizing the services of the METIS system.

5.1. The WorkSense application

The design of the WorkSense application is motivated by a number of studies that have shown the importance of understanding organizational behavior. For example, in [2] the authors showed how we can significantly increase work performance by understanding the face-to-face interactions between individual workers and the formation of various social groups within an organization. We exploited the METIS framework and a number of sensing modalities that were available in our office environment to design *WorkSense*: an application that aims to infer the collaborations and meetings of users, and offers awareness on the impact social interactions may have on their work.
More specifically the application includes the following functional components:

5.1.1. Mobile phone application

The application was implemented on the Android platform and captures the mobility patterns of the users during working hours (visited offices, meeting rooms etc.) and interaction patterns. This information is used to infer social context, mining

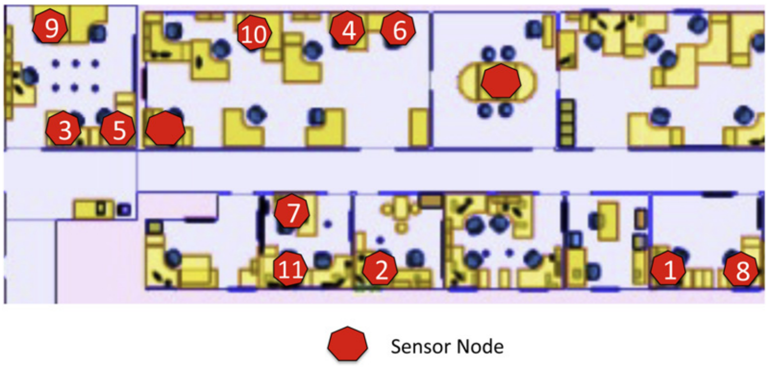

Fig. 16. Sensor nodes were installed on 13 desks including two relay nodes.

working patterns, and to offer awareness to the user. The WorkSense mobile application shows to users details about their collaborations, meetings, and the effect of meetings on their work activities.

5.1.2. *Meeting detection*

A meeting is considered a case where two or more users are co-located for more than a pre-set time threshold (currently 15 min) and they are having a conversation. In the definition, we did not specify a prerequisite that the location should be a meeting room, as we wanted to capture informal meetings that some times can take place in a common room or in the corridor. By incorporating the data captured by the external streams like desktop task loggers, the meeting detection can also illustrate changes in the user's behavior before and after a particular meeting. Such information can allow the user to identify meetings that have a significant impact on their work, or people that may have a profound effect on their performance.

5.1.3. *Collaboration detection*

Although people in our research institution are typically formed into groups that may collaborate within the context of a particular research project, the WorkSense application attempts to capture a more objective picture of how collaborations are taking place within the research lab. Sometimes a user who is not officially assigned to a project can play an important role. Collaboration detection is achieved by applying a community detection algorithm over the co-location data. However, as there are cases where two or more colleagues may share the same office, a straightforward application of the community detection over the co-location data would result in communities that are heavily biased towards people sharing offices, even if they do not typically collaborate. In order to overcome this issue, we defined an *intended visit* as the case where a person visits another person with the intention of having a conversation ($deskEmpty(u_1) \wedge colocated(u_1, u_2) \wedge conversation(u_1)$). If sensing infrastructure is unavailable, then we do not consider the desk sensor information. The result of an intended visit is a directional edge that links the two users where u_1 is the initiator and u_2 is the target. A community detection algorithm is then applied over the directed graph to detect collaborations, which we describe in the next subsection.

From a design perspective, the operation of WorkSense *does not require* the presence of sensors embedded in the environment; however, when such infrastructure is available, the energy performance of the application can be dramatically improved and the accuracy of the gathered information can be increased as well.

5.2. *Deployment*

We deployed the METIS system and the WorkSense application in a working environment involving 11 users for about a month. During this period each user carried an Android phone (Samsung Galaxy S or HTC Desire). The mobile application was able to utilize existing sensing infrastructure that was deployed in our institution (Fig. 16). This included desk occupancy sensors, Bluetooth sensors, and conversation detection infrastructure (see Section 2).

5.2.1. *Energy impact analysis*

We evaluated the energy performance of METIS during the deployment. To compare the difference in battery life *with* and *without* using the sensing offloading, we used pairs of Samsung Galaxy S phones, carried by the same user. One of the phones was set to never offload data (pure phone sensing with Wi-Fi switched off) while the other followed the threshold based offloading scheme. This functionality was swapped between the two phones over consecutive measurements in order to reduce the impact that minor differences of the phone batteries may have on the results. The phones were allowed to discharge only during office hours, by switching them off overnight. During the experiment the phones were not used for any other activities. We then also measured the battery life of the phone when using: local phone sensing with Wi-Fi switched on, sensing disabled with Wi-Fi switched on, and sensing disabled with Wi-Fi switched off. In all these cases, a battery monitor was always running on the phone to measure the battery discharge. We measured the battery level using the *BatteryManager*

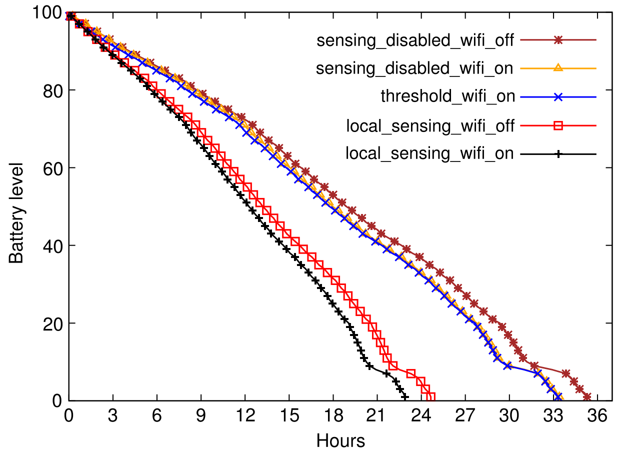

Fig. 17. Battery drain of Samsung Galaxy S phone for various scenarios.

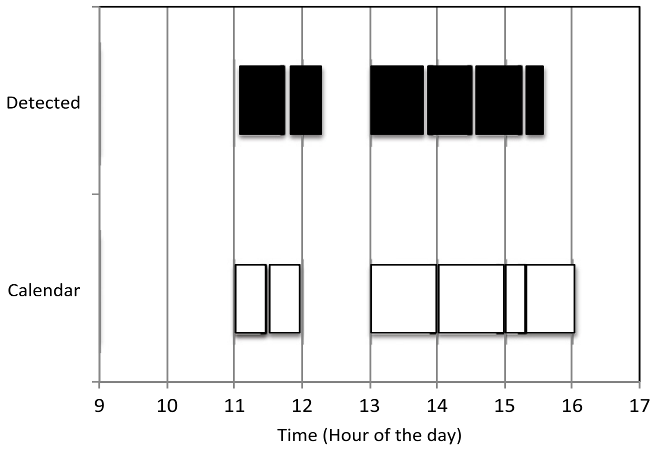

Fig. 18. Timeline of detected vs. calendar meetings.

API of the Android platform. The average consumption over multiple runs is shown in Fig. 17. The results show that, when using the gain threshold scheme and exploiting the sensing infrastructure, the battery lasts up to 45% longer compared to the case of local phone sensing with Wi-Fi on, and 35% longer compared to the case with the Wi-Fi off. Furthermore, the energy cost of using METIS is very close to a mobile phone with the Wi-Fi on and no sensing, and only 6% less compared to a phone that uses no sensing and no Wi-Fi. This is an indication that opportunistic sensing offloading can improve the support for the long-term deployment of sensing applications, by significantly minimizing the impact on the phone's battery life.

5.2.2. Meeting detection analysis

One of the primary motivations behind the design of WorkSense was the fact that social sensing has the potential to capture views over the interactions and activities at the workplace. The most expected result of WorkSense was the detection of formal and informal meetings. The *Meeting Detection* service was able to detect the exact times that meetings are taking place in the laboratory. Furthermore, by utilizing the calendar information such meetings were populated with appropriate meta-data. Fig. 18 shows a snapshot of a timeline where the calendar schedule is contrasted with the actual meetings as they were detected by WorkSense. In this particular snapshot we see how a person that has consecutive meetings can slightly adapt the schedule based on the duration of previous meetings. In this case many meetings were moved earlier: this was not something updated in the calendar. Furthermore, WorkSense was able to capture a number of unscheduled informal meetings typically taking place at the workplace. Although the impact of such meetings may not be immediately obvious, raising awareness to the users about such meetings may prove to be a valuable tool.

5.2.3. Activity type and user specific analysis

WorkSense attempts to offer awareness to the users about their work activity habits and patterns. As an example, information about what activities are mostly taking place at specific times of the day can be detected: an example is shown in Fig. 19 where the detected conversations are shown. Furthermore, interesting results are obtained by comparing the work

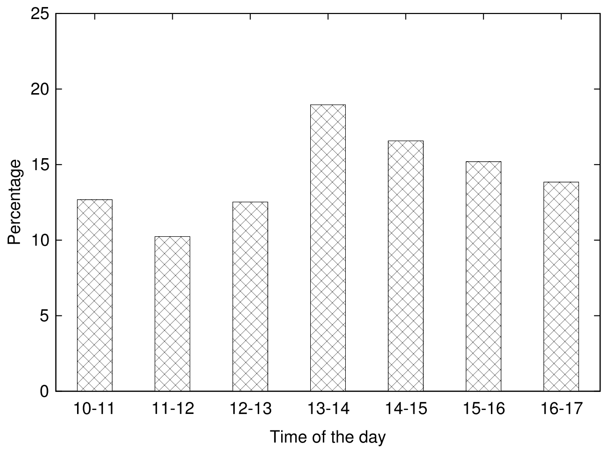

Fig. 19. Conversation patterns with respect to time of the day.

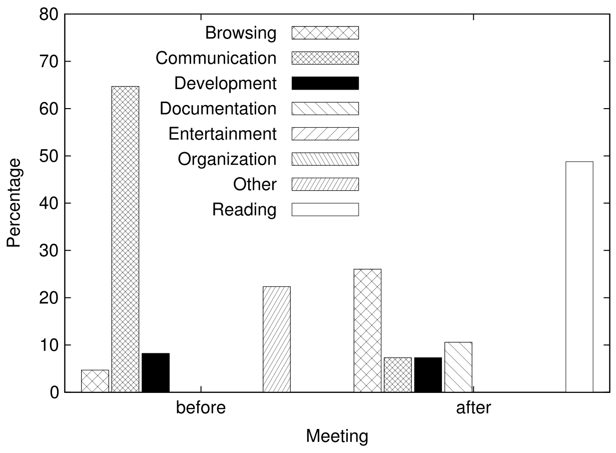

Fig. 20. Work pattern of User 7 before and after a specific meeting.

activities of a user before and after particular meetings. To capture the work activities that act as a ground-truth to show the varying level of the effect of different meetings, we installed task-loggers on the user's computers. The task-loggers periodically log the details about all the applications running on the user's computer, and also detect the current active application. The active window is obtained using an Apple Script module on Mac OS, and `xwininfo` command on Linux. The details about all the running applications are obtained using the `top` command that gives the list of processes running on the user's computer. The meetings detected by the WorkSense can be fused with the data from the task-loggers in the back-end server to extract interesting results: In Figs. 20, 21, it can be seen how the type of activities of User 7 changes drastically after two different meetings.

5.2.4. Community detection analysis

Fig. 22 shows the different groups and project teams as they were reported by the users. We used WorkSense to automatically detect such communities. To do this, we first create a weighted graph of detected collaborations where a link between two users represent the amount of time they spent in meetings and discussions. Afterwards, we apply the *Louvain's* community detection algorithm [17] to detect groups and projects that users belonged to. The Louvain method is a hierarchical greedy algorithm and operates in two phases, repetitively: first, it searches for small communities, and then it combines nodes of the same community and forms a new network. The method uncovers hierarchies of communities and allows to further identify sub-communities, sub-sub-communities etc. This algorithm generates team formations on varying levels of granularity: at the lowest level (Level 1) it identifies smaller communities and at higher levels (e.g., Level 2) it merges smaller communities to form larger clusters. The thicknesses of edges in the graph represent the weight/strength of the link, and the color of a node represents its community. Fig. 23 shows the Level 2 communities where WorkSense was able to identify the separation between people belonging to different departmental groups. The spatial placement of nodes is generated based on a spring model: the larger the spatial difference between two nodes the lesser the collaboration/communication between them. More interestingly though, the output of the community detection when

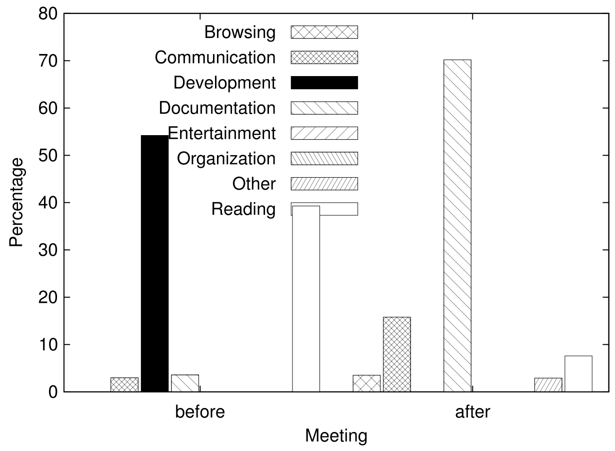

Fig. 21. Work pattern of User 7 before and after another meeting.

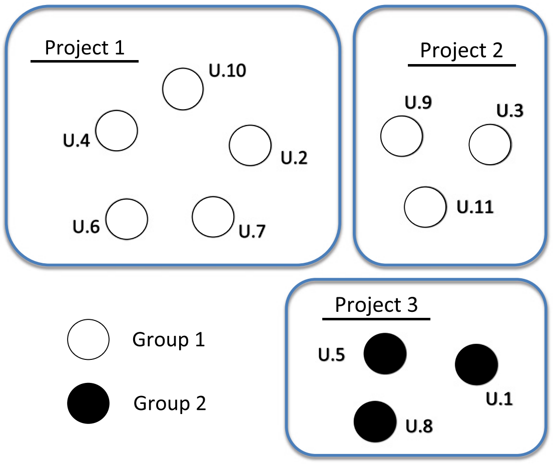

Fig. 22. Communities as reported by the participants of our study.

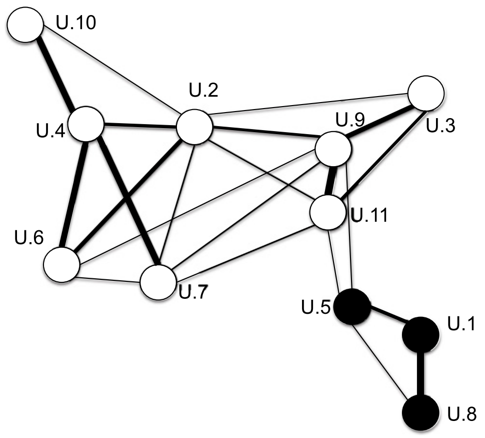

Fig. 23. Automatic detection of groups (Level 2 communities).

operating at Level 1 (Fig. 24), was a more fine grained breakdown of groups that were not related to the information that the users offered but to the projects that users have been working on, even within the same group. Furthermore, the WorkSense application was able to identify cross team collaborations and pinpoint people acting as *bridges* extending collaboration links across teams.

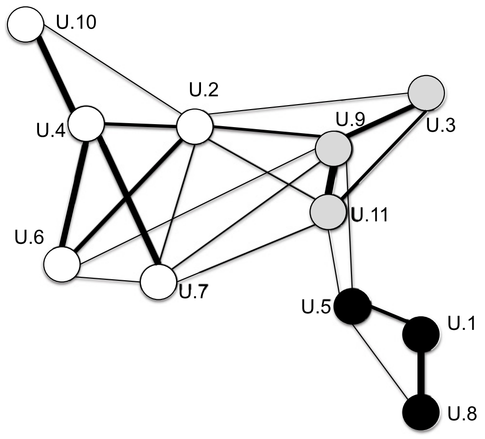

Fig. 24. Automatic detection of projects (Level 1 communities).

6. Related work

Although, energy efficiency has been one of the key design considerations in a number of mobile sensing systems [1,4,5,18] most of them consider only the case of pure phone sensing. A number of systems attempted to address energy consumption issues by using adaptive sensor sampling techniques [5] allowing them to improve energy efficiency by trading-off accuracy. As shown in Section 4, METIS can complement adaptive sensing systems offering further improvements in energy utilization.

A number of systems rely on custom devices for social sensing: the Mobile Sensing Platform (MSP) [19] and Sociometer [20]. Several studies were performed using the Sociometer in order to understand organizational behavior [2]. Since these systems are not based on smartphones, long-term studies are quite difficult to achieve as it might be impractical to ask users to carry external devices for prolonged periods or users may need to be incentivized [21] which again may prove costly.

There have been much work focused on increasing the efficiency of mobile systems by offloading computations [22,23]. CloneCloud [22] enables mobile applications running in an application layer virtual machine (such as the Java Virtual Machine, the Dalvik Virtual Machine on the Android Platform, and the Microsoft's.NET platform) to offload part of the execution to device clones running in the cloud to make the execute time faster and also energy efficient. The authors of [23] present MAUI, a system that achieves energy efficiency through fine-grained code offloading to the infrastructure while minimizing the required code-level changes to applications. These works address the problem of offloading computation to cloud or computing infrastructure to achieve energy efficiency and better performance whereas the proposed system addresses a different problem of sensing task offloading.

With respect to using sensors in the infrastructure: FollowMe [24] lets mobile applications exploit the sensors like cameras and microphones that are available in the environment, for richer context detection. ErdOS [25] is a mobile operating system that extends the battery life of mobile handsets by managing resources proactively and by exploiting opportunistic access to resources in nearby devices using social connections among users. Comparing with FollowMe and ErdOS, our system opportunistically offloads the sensing tasks to the infrastructure to maximize the energy gain while considering the sensing parameters and the user's mobility. CoMon [26] is a platform that aims to increase energy savings by employing heuristics to detect mobile phones that will remain in proximity for a long period, and designing cooperation plans for the mutual benefit of the phones involved. Remora [27] is a platform for assisting sensor sharing among neighboring sensor networks to achieve energy savings while improving accuracy. They too address the problem of sensing offloading, however, the idea has been applied to the domain of Body Sensor Networks (BSNs). None of these works that exploit the sensing infrastructure or nearby sensors provided a solution to the problem of *when to offload smartphone sensing to infrastructure*. In [28] the authors use Bluetooth names to interact with the smart environments, i.e., users are allowed to access content by changing their Bluetooth device names. In [29], the authors propose that mobile phones can serve as data mules for sensor networks due to their ubiquity and show that opportunistic mulling is suitable for office-based deployments. Even though this work involves interaction of mobile phones with sensor networks, it addresses a very different problem than METIS.

Some works [30] deployed sensor networks at the workplace and domestic environments to understand usage patterns such as light use, temperature and motion. These approaches rely on full instrumentation of a building that maybe again costly or impractical. Dynamic power management (DPM) techniques [31] achieve energy efficiency by employing techniques such as selectively turning off idle components and switching between high and low power modes. However, the problem addressed and the solutions provided in our work are very different from the DPM techniques. The work presented in the current article is an extension of our work presented in [32]. We have added several extensions to the current article such as more results in Section 2 on time spent by users in various location types, details about possible extensions to the gain threshold scheme (Section 3), and results on the conversation patterns of the study participants and impact of meetings on work patterns of users (Section 5).

7. Conclusions

In this paper we presented METIS, a mobile sensing platform that supports long-term deployment of social sensing applications by leveraging both phone sensors and sensors in the environment. The system implements a novel sensing distribution scheme that is able to switch between phone and remote sensors considering the various sensing parameters, and mobility patterns of the users. We showed through several benchmark experiments on real traces that the system is able to achieve significant energy savings compared to pure phone sensing and remote sensing schemes. We also reported on a real deployment of METIS through a social application. We showed that the application is able to infer the collaborations and meetings of the users while achieving a battery lifetime that is only 6% shorter in duration than a mobile phone that performs no sensing. We plan to extend the system by supporting additional sensing modalities and also enhance the application to recommend effective work behaviors to people at the workplace.

Acknowledgments

The authors would like to thank the members of the Systems Research Group at the University of Cambridge for their invaluable feedback, and the participants of the social experiment. This work was supported through the Gates Cambridge Trust and EPSRC grant EP/G069557/1.

References

[1] A. Campbell, S. Eisenman, N. Lane, E. Miluzzo, R. Peterson, H. Lu, X. Zheng, M. Musolesi, K. Fodor, G.-S. Ahn, The rise of people-centric sensing, IEEE Internet Computing (2008) 12–21.
[2] W. Lynn, B. Waber, S. Aral, E. Brynjolfsson, A. Pentland, Mining face-to-face interaction networks using sociometric badges: predicting productivity in an IT configuration task, in: ICIS'08, 2008.
[3] B. Kim, J.-Y. Ha, S. Lee, S. Kang, Y. Lee, Y. Rhee, L. Nachman, J. Song, Adnext: a visit-pattern-aware mobile advertising system for urban commercial complexes, in: HotMobile'11, ACM, 2011.
[4] K.K. Rachuri, M. Musolesi, C. Mascolo, P. Rentfrow, C. Longworth, A. Aucinas, EmotionSense: a mobile phones based adaptive platform for experimental social psychology research, in: UbiComp'10, ACM, 2010.
[5] H. Lu, J. Yang, Z. Liu, N. Lane, T. Choudhury, A. Campbell, The Jigsaw continuous sensing engine for mobile phone applications, in: SenSys'10, ACM, 2010.
[6] H. Lu, A. Bernheim Brush, B. Priyantha, A. Karlson, J. Liu, SpeakerSense: energy efficient unobtrusive speaker identification on mobile phones, in: Pervasive'11, Springer, 2011.
[7] Hidden Markov Model Toolkit. http://htk.eng.cam.ac.uk.
[8] C. Efstratiou, I. Leontiadis, M. Picone, K.K. Rachuri, C. Mascolo, J. Crowcroft, Sense and sensibility in a pervasive world, in: Pervasive'12, 2012.
[9] S. Dawson-Haggerty, X. Jiang, G. Tolle, J. Ortiz, D. Culler, sMAP: a simple measurement and actuation profile for physical information, in: Sensys'10, ACM, 2010.
[10] R. Friedman, A. Kogan, Y. Krivolapov, On power and throughput tradeoffs of wifi and bluetooth in smartphones, in: INFOCOM, IEEE, 2011.
[11] L. Zhang, B. Tiwana, Z. Qian, Z. Wang, R.P. Dick, Z.M. Mao, L. Yang, Accurate online power estimation and automatic battery behavior based power model generation for smartphones, in: CODES/ISSS'10, ACM, 2010.
[12] A. Rice, S. Hay, Decomposing power measurements for mobile devices, in: PerCom'10, IEEE, 2010.
[13] A. Kansal, S. Nath, J. Liu, F. Zhao, SenseWeb: an infrastructure for shared sensing, IEEE MultiMedia 14 (4) (2007) 8–13.
[14] P. Bahl, V. Padmanabhan, RADAR: an in-building RF-based user location and tracking system, in: INFOCOM'00, IEEE, 2000.
[15] A. Haeberlen, E. Flannery, A.M. Ladd, A. Rudys, D.S. Wallach, L.E. Kavraki, Practical robust localization over large-scale 802.11 wireless networks, in: MobiCom'04, ACM, 2004.
[16] K.K. Rachuri, M. Musolesi, C. Mascolo, Energy-accuracy trade-offs in querying sensor data for continuous sensing mobile systems, in: Mobile Context-Awareness Workshop'10, 2010.
[17] V.D. Blondel, J.-L. Guillaume, R. Lambiotte, E. Lefebvre, Fast unfolding of communities in large networks, Journal of Statistical Mechanics: Theory and Experiment (2008) P10008.
[18] Y. Wang, J. Lin, M. Annavaram, Q. Jacobson, J. Hong, B. Krishnamachari, N. Sadeh, A framework of energy efficient mobile sensing for automatic user state recognition, in: MobiSys'09, ACM, 2009.
[19] T. Choudhury, G. Borriello, S. Consolvo, D. Haehnel, B. Harrison, B. Hemingway, J. Hightower, P.P. Klasnja, K. Koscher, A. LaMarca, J.A. Landay, L. LeGrand, J. Lester, A. Rahimi, A. Rea, D. Wyatt, The mobile sensing platform: an embedded activity recognition system, IEEE Pervasive Computing 7 (2) (2008) 32–41.
[20] T. Choudhury, A. Pentland, Sensing and modeling human networks using the sociometer, in: ISWC'03, 2003.
[21] M. Musthag, A. Raij, D. Ganesan, S. Kumar, S. Shiffman, Exploring micro-incentive strategies for participant compensation in high-burden studies, in: Ubicomp'11, ACM, 2011.
[22] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, A. Patti, CloneCloud: elastic execution between mobile device and cloud, in: EuroSys'11, ACM, 2011.
[23] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, P. Bahl, MAUI: making smartphones last longer with code offload, in: MobiSys'10, ACM, 2010.
[24] B. Greenstein, B. Longstaff, FollowMe: enhancing mobile applications with open infrastructure sensing, in: HotMobile'11, ACM, 2011.
[25] N. Vallina-Rodriguez, J. Crowcroft, ErdOS: achieving energy savings in mobile OS, in: MobiArch'11, ACM, 2011.
[26] Y. Lee, Y. Ju, C. Min, S. Kang, I. Hwang, J. Song, CoMon: cooperative ambience monitoring platform with continuity and benefit awareness, in: MobiSys'12, ACM, 2012.
[27] M. Keally, G. Zhou, G. Xing, J. Wu, Remora: sensing resource sharing among smartphone-based body sensor networks, in: IWQoS'13, IEEE, 2013.
[28] N. Davies, A. Friday, P. Newman, S. Rutlidge, O. Storz, Using bluetooth device names to support interaction in smart environments, in: Mobisys'09, ACM, 2009.
[29] U. Park, J. Heidemann, Data muling with mobile phones for sensor nets, in: Sensys'11, ACM, 2011.
[30] J. Lifton, M. Feldmeier, Y. Ono, C. Lewis, J.A. Paradiso, A platform for ubiquitous sensor deployment in occupational and domestic environments, in: IPSN'07, ACM, 2007.
[31] L. Benini, A. Bogliolo, G.D. Micheli, A survey of design techniques for system-level dynamic power management, IEEE Transactions on VLSI Systems 8 (2000) 299–316.
[32] K.K. Rachuri, C. Efstratiou, I. Leontiadis, C. Mascolo, P.J. Rentfrow, METIS: exploring mobile phone sensing offloading for efficiently supporting social sensing applications, in: PerCom'13, IEEE, 2013.