SociableSense: Exploring the Trade-offs of Adaptive Sampling and Computation Offloading for Social Sensing

Kiran K. Rachuri, Cecilia Mascolo
Computer Laboratory
University of Cambridge
{kkr27, cm542}@cam.ac.uk

Mirco Musolesi
School of Computer Science
University of Birmingham
m.musolesi@cs.bham.ac.uk

Peter J. Rentfrow
Faculty of Politics, Psychology,
Sociology and International Studies
University of Cambridge
pjr39@cam.ac.uk

ABSTRACT

The interactions and social relations among users in workplaces have been studied by many generations of social psychologists. There is evidence that groups of users that interact more in workplaces are more productive. However, it is still hard for social scientists to capture fine-grained data about phenomena of this kind and to find the right means to facilitate interaction. It is also difficult for users to keep track of their level of sociability with colleagues. While mobile phones offer a fantastic platform for harvesting long term and fine grained data, they also pose challenges: battery power is limited and needs to be traded-off for sensor reading accuracy and data transmission, while energy costs in processing computationally intensive tasks are high.

In this paper, we propose *SociableSense*, a smart phones based platform that captures user behavior in office environments, while providing the users with a quantitative measure of their sociability and that of colleagues. We tackle the technical challenges of building such a tool: the system provides an adaptive sampling mechanism as well as models to decide whether to perform computation of tasks, such as the execution of classification and inference algorithms, locally or remotely. We perform several micro-benchmark tests to fine-tune and evaluate the performance of these mechanisms and we show that the adaptive sampling and computation distribution schemes balance trade-offs among accuracy, energy, latency, and data traffic. Finally, by means of a social psychological study with ten participants for two working weeks, we demonstrate that SociableSense fosters interactions among the participants and helps in enhancing their sociability.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Wireless Communication; H.1.2 [**User/Machine Systems**]: Human Information Processing; H.3.4 [**Systems and Software**]: Distributed Systems; J.4 [**Social and Behavioral Sciences**]: Psychology, Sociology

General Terms

Algorithms, Design, Experimentation.

MobiCom'11, September 19–23, 2011, Las Vegas, Nevada, USA.

Keywords

Computation Offloading, Energy-Accuracy Trade-offs, Mobile Phone Sensing, Social Interactions, Social Psychology.

1. INTRODUCTION

The study of interactions and sociability among users in workplaces has been of interest to social scientists for many generations. A deep understanding of social dynamics in the office environments helps companies to better manage employees and increase their productivity. There is substantial evidence that groups in which individuals are more socially connected at work are also more productive [23, 27, 32]. Interactions are heavily influenced by the colocation in the same office spaces and the type of work performed by the employees. For this reason, there is generally a non-uniform distribution of the strength of these social interactions. Social scientists study user behavior and interactions in workplaces mostly through self-reports, however, these are found to be biased towards positive (or pleasant) interactions [28]. In general, participants find it hard to accurately remember and report details. Moreover, incentives and suggestions about possible further interactions are usually given post-analysis and through costly meeting sessions with the workers.

Smart phones represent an ideal computing platform to conduct social psychological studies in the office environments and to *close the loop* by providing feedback to the users as they are unobtrusive and sensor rich platforms. Recently, we have seen a soaring number of systems based on smart phones aimed at social psychological and behavioral monitoring studies [16, 24, 31]. Most of these systems are useful to collect data for the social scientists. There is an opportunity to use these data to provide help, services, or simply useful information to participants. However, the data harvesting through mobile phones still presents a variety of challenges to developers: energy consumption is still very high for both data transmission and resource-intensive local computation. Adaptive sampling of the sensors to preserve energy, and the data offloading to servers for remote computing of the analysis need to be traded-off for energy preserving on the phones, which still need to be functional to the users.

In this paper, we present the design, implementation, performance evaluation, and deployment of *SociableSense*, a smart phones based sensing platform with the aim of providing real-time feedback to users in order to help them in fostering their interactions and improving their relations with colleagues. SociableSense implements novel solutions in terms of adaptive sensor sampling and intelligent distributed computation based on the context and phone status. More

specifically, a *sensor sampling component* adaptively controls the sampling rate of accelerometer, Bluetooth, and microphone sensors while balancing energy-accuracy-latency trade-offs based on *reinforcement learning* mechanisms [5, 18]. A *computation distribution component* based on *multi-criteria decision theory* [19] dynamically decides where to perform computation of tasks such as data analysis and classification, by considering the importance given to each of the dimensions: energy, latency, and data sent over the network. Finally, with respect to the support for the application itself, a *social feedback component* determines the sociability of users (i.e., a quantitative measure of the quality of their relationships) based on interaction and colocation patterns extracted from the sensed data at run-time and provides them with feedback about their sociability, strength of relationship with colleagues, and also alerts about opportunities to interact.

We perform a two-part evaluation of the SociableSense system. First, we perform several micro-benchmark tests to fine-tune and evaluate the various components. Second, in order to demonstrate the usefulness of SociableSense to the social scientists and participants, we conduct a social psychological study in an office environment where 10 participants carried mobile phones for two working weeks. To the best of our knowledge, SociableSense is the first mobile sensing system that implements sensor sampling that is adaptive in real-time with respect to the changing context, and a computation distribution scheme that takes into account the changing resources and the requirements of the researchers collecting the sensor data. Furthermore, we believe that this is the first work that brings *mobile sensing* and *social network theory* together in order to provide real-time feedback to users about their relationships in an effective, and, at the same time, efficient way.

More specifically, the key contributions of our work can be summarized as follows:

- We illustrate a novel sensor sampling scheme based on reinforcement learning that adaptively controls the rate at which sensors in the phone are sampled, and show that it achieves better accuracy and latency without considerably compromising on the energy consumption.
- We present a computation distribution scheme that dynamically decides where to perform the computation of classification tasks, i.e., locally or on remote servers including cloud services, while considering three dimensions: energy, latency, and data traffic. We also present a rule-based framework through which experiment designers can adapt the behavior of this scheme at run-time.
- We present the design, implementation, and evaluation of a system for quantitatively measuring the sociability and relations of users in the office environments through our adaptive framework. We also *close the loop* by providing real-time feedback about their *sociability*, the strength of their relations with colleagues, and opportunities to interact. By inferring the *most sociable person* in the office, we provide implicit incentives to users to become more *sociable*.

The remainder of this paper is organized as follows: we describe the motivation of the work in Section 2, and our approach in Section 3. We then present the details of our system in Section 4. We present the evaluation through micro-benchmarks in Section 5 and through a social experiment in Section 6. We then present the related work in Section 7, and, finally, we conclude the paper in Section 8.

2. RELATIONS IN WORKPLACES

Social science researchers have been devoting their attention on investigating behavior and interactions of users in the workplaces for a long time, trying to answer questions about different aspects of the problem. *What are the interaction and colocation patterns among users in the office or corporate environments? Do people socialize more in personal office spaces or in common spaces like coffee rooms? Which work-group members socialize with one-another and why?* Considering evidence that work-groups with highly connected team members outperform less connected ones [23, 27, 32], the importance of these research questions is clear: Understanding how work-groups socialize naturally and identifying the factors that moderate socialization could have significant effects on group performance and productivity.

Moreover, some of the questions concerning the users themselves include: *Who is the most sociable person in the group? Do I socialize more with person X or person Y ? Could I get to know X more if I get a chance to interact with him/her in the coffee room every day? How does my sociability vary with respect to the time of day or day of week?*

Investigations of workplace socializing tend to rely primarily on self-report methods. Although such methods can be useful and effective for assessing attitudes, beliefs, and emotions, considerable evidence suggests that reports of behavior tend to be inaccurate [28]. Laboratory based studies enable researchers to observe behavior instead of relying on behavioral reports, however, the nature of laboratory studies can make participants' experiences unnatural and contrived. Moreover, using these methods, real-time feedback/alerts about users' own relations are hard to provide. It is also difficult for participants to know precise answers to the above questions, which go beyond their own approximate observations.

Smart phones can prove invaluable for the long term and continuous monitoring of people in their environment. Since everyone has a mobile phone, they are a perfect platform for conducting behavioral monitoring studies. Moreover, they are also very effective to *close the loop* by providing feedback and alerts to participants. However, before smart phones can be thought as the perfect platform for social studies, there are many challenges which need to be addressed: first, since mobile phones are battery powered, the sensor sampling rate should be adaptively controlled while considering not only accuracy, energy, but also latency, as timely alerts for socializing opportunities are important: phones are primarily to be used by users for their tasks and energy should not be depleted. Second, efficient classifiers have to be developed to derive accurate inferences from the raw data of potentially inaccurate sensors of mobile phones. Third, devices should be able to effectively use the local and cloud resources not only to achieve energy efficiency but also to reduce the data transmission over the network in order to avoid to use the data plans of participants. Fourth, with respect to sensing social interactions, efficient algorithms have to be designed to capture the relations among users. Fifth, online feedback mechanisms have to be designed in order to provide timely alerts to users (e.g., about opportunities to interact with their colleagues).

3. OUR APPROACH AT A GLANCE

In this section, we describe how we address the key challenges in building the SociableSense system.

Adaptive Sensor Sampling The sensors embedded in the mobile phones have to be sampled continuously in order to capture user behavior, however, this leads to the depletion of battery. If the sensors are sampled at a slower rate, then it may not be possible to capture the user behavior accurately. Therefore, we design an adaptive sampling technique that balances the energy-accuracy-latency trade-offs through the use of *linear reward-inaction learning* [5, 18] that is based on the theory of *learning automata* [26]. The adaptive sampling scheme adjusts the sampling rate of these sensors dynamically based on the context of the user in terms of events observed (interesting or not) and thereby achieves an improved accuracy and latency without considerably compromising on the energy consumption of the system, i.e., the sensors are sampled at a high rate when there are *interesting* events observed and at a low rate when there are no events of interest. We provide detailed explanation about this adaptation scheme in Section 4.1.

Intelligent Computation Distribution Once the data from the sensors are sampled, they need to be processed to derive high level inferences. This processing might be trivial in terms of resource consumption for some classification tasks like detecting whether a person is stationary or moving, and intensive for some other tasks like speaker identification from microphone data or image recognition from camera sensing. Mobile phones have limited computing power but they can depend on remote computation performed on back-end servers such as cloud farms. However, in general, data transmission is costly in terms of energy consumption and not all users may have unlimited data plans. Therefore, the allocation of the execution of computational tasks is of key importance in such systems. We design a computation distribution scheme based on *multi-criteria decision theory* [19] that decides whether to perform the computation locally on the phone or remotely in the cloud by considering various dimensions such as energy, latency, and data sent over the network. This scheme smartly distributes the classification tasks among local and cloud resources while balancing the energy-latency-traffic trade-offs. Furthermore, we also design a rule based framework for dynamically adapting the behavior of this distribution scheme with respect to decrease/increase in the mobile phone resources (like battery charge/discharge cycles, user's data plan running out of allowance). We provide details about this component in Section 4.2.

Measuring Sociability for Effective Feedback Based on the inferences derived from the classifiers, we quantitatively measure the *sociability* of users and the strength of relations with their colleagues using the so-called *network constraint* [6]. The quantification of sociability is performed with respect to user colocation and interaction patterns. In order to enhance the sociability of users, we design alert and feedback mechanisms that inform the users about *opportunities* to interact with colleagues in social locations and relationships that need attention. We explain these mechanisms in detail in Sections 4.3 and 4.4.

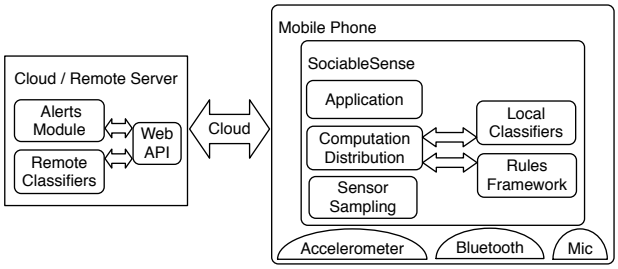


Figure 1: Architecture of the SociableSense system.

4. SYSTEM COMPONENTS

In this section we describe the details of the components of the SociableSense system, focusing on the description of its novel underlying mechanisms. The overall system architecture of SociableSense is shown in Figure 1.

4.1 Sensor Sampling

This component is responsible for adaptively querying the data from the accelerometer, Bluetooth, and microphone sensors while balancing energy-accuracy-latency trade-offs. We first present the learning technique based on the theory of learning automata [26] that is used for adaptive sampling.

4.1.1 Design Methodology

After raw data are queried from the sensors, they are sent to the classifiers to derive high level inferences. We refer to these inferences as *events*. For example, the X, Y, Z axes values are queried from the accelerometer sensor continuously for a certain period of time and are then sent to the activity recognition classifier in order to extract the *moving* or *stationary* events. We further categorize each of these classified events into two classes, *viz.*, *missable* and *unmissable*. A *missable* event indicates that no interesting external phenomenon has happened and the corresponding sensor can sleep during this time. An *unmissable* event is an event of interest observed in the environment that should not be missed by the sensor. The classification of an event as missable or unmissable is application dependent. For example, a categorization of missable and unmissable events for voice recognition classifier based on the microphone sensor can be as follows: *silence* might be missable, *some audible data* might be unmissable.

4.1.2 Learning based Adaptation

The system uses a learning technique based the theory of learning automata [26] to control the sampling rate of the sensors. In particular, we use the *linear reward-inaction* algorithm [5, 18] because of its suitability for adaptively adjusting the sampling rate of sensors[1]. Learning automata based techniques are defined in terms of *actions*, *probability* of taking these actions, and their resulting *success* or *failure*. In the context of learning automata, there is only one action in our scenario, i.e., sensing from a sensor. The decision whether to sense or not from a sensor is based on a probability value, which we call *probability of sensing*. The sensing action results in either *success* or *failure*, which are defined

[1]Alternative techniques include information theoretical approaches such as compressed sensing [12], which are orthogonal with respect to our application-level approach.

as follows: when data sensed from a sensor corresponds to an unmissable event (for example, when an audio sample is recorded through the microphone sensor, and it contains some audible data) then we term this as *success*. Similarly, when data sensed from a sensor corresponds to a missable event (for example, audio data from microphone sensor contains no audible data and only silence) then we term this as *failure*. The idea is that when a sensor expends some energy in sensing and this results in capturing an event of interest then it is considered as a *success*, otherwise as a *failure*. The probability of sensing from a sensor is dynamically adjusted according to their previous successes and failures. The technique works as follows: let p_i be the probability of sensing from a sensor s_i where i={*accelerometer, Bluetooth, microphone*}, and a_i be the sensing action on a sensor s_i.

If the sensing action a_i results in an unmissable event, it means that the system has detected an interesting event (i.e., a *success*), so that probability is increased according to the following formula:

$$p_i = p_i + \alpha(1 - p_i),\ where\ 0 < \alpha < 1 \tag{1}$$

If the sensing action a_i results in a missable event, it means that the system has not detected an interesting event (i.e., a *failure*), so that probability is decreased according to the following formula:

$$p_i = p_i - \alpha p_i,\ where\ 0 < \alpha < 1 \tag{2}$$

Therefore, by adopting these mechanisms, the sampling rate adapts to the context of the user. We also limit the lower and upper bounds of these probabilities to 0.1 and 0.9, respectively. If the probability of sensing is less than 0.1 then the duty cycling may fall below 10%, and if it is more than 0.9, the duty cycling may be over 90%. We limit the lower bound to avoid having a very small sampling rate, which will potentially lead to miss important events. At the same time, we limit the upper bound to avoid a too aggressive sampling, which reduces the battery life. We present the evaluation of the learning technique and compare its performance with various static functions and dynamic adaptation techniques in Section 5.1.

4.2 Computation Distribution

After raw data are queried from the sensors, the sensor sampling component sends them to the computation distribution component, which is responsible for extracting the high level inferences from the sensed data by performing the computation efficiently using local and cloud resources.

4.2.1 Methodology

We refer to the task of classifying data from a sensor with respect to a classifier as *task* and we assume that *energy*, *latency*, and *total data sent over the network* for each of the tasks are pre-calculated and available to this module beforehand. This is a practical assumption as these values can be calculated for most of the common sensing tasks relatively easily: for example, the Nokia Energy Profiler [13] can be used to obtain these values on the Symbian S60 platform. We also assume that classifiers are preloaded both locally on the phone and remotely in the server/cloud. For certain tasks it is possible to divide a bigger task into smaller sub-tasks, for example, a speaker identification task can be subdivided into extracting the characterizing features from the audio file, and processing them to identify the speaker.

Let T be a task that can be divided into the subtasks $t_1, t_2, t_3, \ldots t_n$ (where $n \geq 1$). If a task cannot be divided into subtasks then we assume that it is composed of one single subtask (i.e., the main task itself – n is equal to 1 in this case). Each subtask t_i can be computed locally on the phone or remotely in the cloud. This results in a total of 2^n unique combinations in which T can be computed. We refer to a combination as a *configuration*. For each configuration c_i (i in $[1, 2, \ldots 2^n]$), let e_i, l_i, d_i be the total energy consumption, latency, and total data sent over the network to process the task (including all subtasks). We use a technique based on *multi-criteria decision theory* [19] to select the configuration for processing the overall task.

We first define a utility function with respect to energy (u_{e_i}) for a configuration c_i as:

$$u_{e_i} = \frac{e_{min} - e_i}{e_i} \tag{3}$$

where e_i is the energy consumption for processing the task T using the configuration c_i and e_{min} is the minimum of $\{e_i, i = 1, 2, \ldots 2^n\}$. This utility function quantifies the advantage of using a configuration over the best configuration with respect to energy consumption, and its range is $[-1, 0]$. This utility function [15, 19] can be used to decide which configuration to use for achieving energy efficiency, and that is indeed the combination with highest utility (or highest gain/advantage).

We also consider other performance metrics such as latency and total data sent over the network. We define utility functions for these performance metrics in a similar fashion.

$$u_{l_i} = \frac{l_{min} - l_i}{l_i} \tag{4}$$

where l_i is the latency for processing the task T using the configuration c_i and l_{min} is the minimum of $\{l_i,\ i = 1, 2, \ldots 2^n\}$.

$$u_{d_i} = \frac{d_{min} - d_i}{d_i} \tag{5}$$

where d_i is the total data sent over the network for processing the task T using the configuration c_i and d_{min} is the minimum of $\{d_i, i = 1, 2, \ldots 2^n\}$. In case of local computation d_i is zero, in cases like this we consider the value of utility function to be zero. The decision on which configuration c_i to use for processing the task should take into account all these performance metrics. For this reason, we define a combined utility function based on the corresponding utilities for each of the performance metrics. The combined utility function (u_{c_i}) for c_i is an additive utility function [14, 20] defined as follows:

$$u_{c_i} = w_e u_{e_i} + w_l u_{l_i} + w_d u_{d_i} \tag{6}$$

where w_e, w_l, w_d are the weights (or importance) given by the experiment designers for energy, latency, and data sent over the network, respectively, such that $w_e + w_l + w_d = 1$. For example, participants with unlimited data plans may not worry about the amount of data sent over the network but maybe concerned about the battery, so the experiment designers may give a higher weight to energy than data sent over network. Finally, the configuration c_i for which u_{c_i} is maximum is used to process the task T.

This scheme can also be generalized to make it applicable to various other scenarios considering additional performance metrics or dimensions, and computation models

(like cloud computation through Wi-Fi, cloud computation through 3G). Let $D_1, D_2, \ldots D_k$ be the k dimensions to be considered, and $M_1, M_2, \ldots M_q$ be the computation models available for selecting a configuration. The total configurations for a task T with n subtasks will be q^n. Then the utility function for these configurations for each dimension would be:

$$u_{D_{ji}} = \frac{D_{jbest} - D_{ji}}{D_{ji}}, \forall\ j \text{ in } [1, 2, .. k],\ \forall\ i \text{ in } [1, 2, .. q^n] \quad (7)$$

where D_{jbest} is the value of the best case scenario for the dimension D_j. The overall utility for each of the configurations can be calculated as:

$$u_{c_i} = \sum_{j=1}^{k} w_j u_{D_{ji}},\ where \sum_{j=1}^{k} w_j = 1,\ \forall\ i \text{ in } [1, 2, .. q^n] \quad (8)$$

where w_j is the weight for the dimension D_j. Finally, the configuration with the maximum utility value, i.e., $C = max\{u_{c_i}, i = 1, 2, \ldots q^n\}$ is used to process the task.

4.2.2 Algorithmic Complexity

The main computational tasks involved in this scheme are: computing the utility functions for each of the performance metrics, and the total utility value for each of the configurations. Each of these procedures has an algorithmic complexity of $O(q^n)$, where n is the total number of subtasks that a given task T can be divided into, and q is 2 as we consider local and remote computation models. Since these procedures are executed one after the other, the overall algorithmic complexity is $O(2^n)$. Even though this is exponential, the total number of subtasks n is in practice small [10, 24]. For example, in case of computationally intensive tasks like speaker recognition n is generally around 2 to 4 [24, 31].

4.2.3 Adaptation of Weights

The designers of the experiments are expected to provide the weights for energy, latency, and total data sent over the network. However, some resources like battery life and total data left in user's data plan change over time. The battery charge of mobile phones lasts for a limited amount of time, and most users have limited data plans and the costs after exceeding this limit are generally high. Therefore, it seems sensible to use different weights for different "states" of the devices. For example, the designers of the experiments might want to give more importance to latency when the battery is full, and when the battery is near depletion, they might want to assign higher priority to energy. Moreover, they might also want to put an upper limit on the amount of data that can be sent over the network per day. We design a framework where experiment designers can add simple rules to switch the weights of the metrics with changing resource levels. The specification of rules is in XML format, and a sample configuration is shown in Figure 2. The weights configured for the condition that matches first will be used. If none of the conditions are satisfied then the weights configured as the default will be used.

```
<rules>
    <condition="battery_left > 80 and data_sent < 50MB">
       <weight metric="energy">33.3</weight>
       <weight metric="latency">33.3</weight>
       <weight metric="data">33.3</weight>
    </condition>
    <condition="battery_left < 20">
       <weight metric="energy">60</weight>
       <weight metric="latency">20</weight>
       <weight metric="data">20</weight>
    </condition>
    <condition="data_sent > 50MB">
       <weight metric="energy">20</weight>
       <weight metric="latency">20</weight>
       <weight metric="data">60</weight>
    </condition>
    <condition="default">
     <weight metric="energy">33.3</weight>
     <weight metric="latency">33.3</weight>
     <weight metric="data">33.3</weight>
    </condition>
</rules>
```

Figure 2: Sample rules for adaptation of weights for energy, latency, and data sent over the network.

4.3 Sociability Measurements

We define *sociability* of a user as the strength of the user's connection to his/her social group. In other words, this metric is used to represent the quantity and quality of his/her relationships with the colleagues. We measure the strength of a user's relations and his/her overall sociability based on the *network constraint* [6][2]. In a social network, the network constraint for a node quantifies the strength of the node's connectivity. For any two persons in a social network, the person with lower *network constraint* value is considered to have higher strength in terms of connectivity. The network constraint N_i for a node i in a social network is measured as:

$$N_i = \sum_j (p_{ij} + \sum_q p_{iq} p_{qj})^2, q \neq i, j; j \neq i \quad (9)$$

where p_{ij} is the proportion of time i spent with j, i.e., the total time spent by i with j divided by the total time spent by i with all users in the network.

Based on this concept, we measure the *sociability* of users with respect to the dimensions: colocation and interaction patterns. We define colocation of a pair of users as being in proximity to each other and interaction as speaking to each other. The system captures the colocation patterns of users through the Bluetooth sensor, and the interaction patterns through the microphone sensor and a speaker identification classifier [31][3]. We refer to the network constraint for colocation network as *colocation network constraint* (NC). This is calculated as follows:

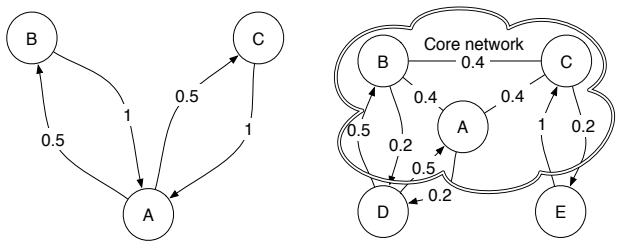
$$NC_i = \sum_j (pc_{ij} + \sum_q pc_{iq} pc_{qj})^2, q \neq i, j; j \neq i \quad (10)$$

where pc_{ij} is the proportion of time user i is colocated with user j. Similarly, we refer to the network constraint for the interaction network as *interaction network constraint* (NI) and is calculated as follows:

$$NI_i = \sum_j (ps_{ij} + \sum_q ps_{iq} ps_{qj})^2, q \neq i, j; j \neq i \quad (11)$$

[2]An alternative and in a way similar measure is the *kith index* [32].

[3]There are other non-verbal modes of interactions among users like gestures however these are hard to capture through the sensors of smart phones.



(a) A is more sociable than B,C. (b) C is more sociable than A.

Figure 3: Sociability measurement examples.

where ps_{ij} is the proportion of time user i has interacted with user j. A smaller network constraint means smaller ps_{ij} and ps_{iq} values which in-turn means that user has spent time interacting with many colleagues, i.e., he/she is more sociable.

In a social network of n nodes, the strength of relations of a user i with respect to colocation is calculated based on the pc_{ij} where $j = 1, 2, \ldots n, j \neq i$ and with respect to interactions is calculated based on the ps_{ij} where $j = 1, 2, \ldots n, j \neq i$. We call the user i where $i = min\{NC_k,\ k = 1, 2, \ldots n\}$ (i.e., user with least colocation network constraint value) the *mayor* of the group with respect to colocation, and similarly the user j where $j = min\{NI_k,\ k = 1, 2, \ldots n\}$ (i.e., user with least interaction network constraint value) the *mayor* of the group with respect to interactions. The alerts about the *mayors* are sent to all mobile phones periodically to encourage active participation of the users in the experiment and also to motivate them to socialize more by adding the *competition* and *gaming* factors.

We use two examples to demonstrate the quantification of sociability through the network constraint. Figure 3(a) shows a social network of three nodes. Let us say that A spent 5 hours with B and 5 hours with C, and B, C did not spend any time with each other. The fraction of time A spent with B is 0.5 and with C is 0.5 and these are represented as weights in the graph. However, B has spent time only with A, therefore the weight of the edge (B, A) is one, similarly weight of edge (C, A) is one too. The ranking with respect to decreasing order of sociability (or increasing order of network constraint) is: $A(0.5)$, $B(1.25)$, $C(1.25)$. Since A distributes his/her time among two contacts, he/she is less constrained compared to B and C, therefore A is more sociable. Figure 3(b) shows a social network of five nodes. Let us assume that there is a core social network consisting of nodes A, B, and C, and two new persons D and E have recently joined the network. D has shared his interactions with two people in the network, whereas E has connected only to one person. Therefore, D is connected more strongly than E. The constraint for D and E are 1.13 and 1.32, respectively, which makes D more sociable than E. Further, A and C are both connected to three nodes, however, C's constraint (0.69) is lesser than A's constraint (0.83) as more information flows through C than A as the graph breaks into two components without C. Therefore, C is considered more sociable than A. This kind of data about the sociability among users is of very high interest to social scientists and corporates as it helps them to understand social patterns of users at a great level of detail and, moreover, it also has impact on their productivity [23, 27, 32].

4.4 Indoor Localization

The indoor localization feature in the SociableSense system is based on the Bluetooth sensor, and is implemented to identify the users in sociable locations to notify colleagues, and to study the influence of type of location on the sociability of users. By placing Bluetooth devices at various locations in the office spaces, we can achieve a coarse grained localization[4]. The main reason to use this methodology is the following: since the system is already required to scan for Bluetooth devices for identifying colocation of the users, we do not need to expend additional energy on sensing from other sensors like Wi-Fi or GPS that are generally expensive in terms of energy consumption. Moreover, Wi-Fi is not available on all smart phones, and GPS does not generally work in indoor office locations.

Our main aim is to differentiate between work and social spaces. The work locations are places where users work and spend most of their office time. Sociable locations are places where users socialize and spend time during breaks like coffee rooms, common rooms, and game rooms. When a user is in a sociable location, alert about this is sent to all other participants, so that interested people can join the user and socialize with him/her.

4.5 Implementation

The system is implemented on the Nokia 6210 Navigator phone using *Python for Symbian S60* (PyS60) [29]. Sensor sampling, computation distribution, and social feedback components run on the mobile phone, while the network constraint calculation module and location alerts generation module run on the remote server. The remote server module is implemented in Python and is deployed on a powerful back-end server (Intel Xeon Octa-core E5506 2.13GHz processor, and 12 GB RAM). The system on the mobile phone connects to this server through a web-based API implemented using the *HTTP Connection* module of PyS60.

The speaker recognition module is based on that used in the EmotionSense system [31] which is implemented using the Hidden Markov Model Toolkit (HTK) [17]. We collect training data (approximately 10 minutes) from each participant of the social psychological experiment (Section 6) and generate a background Gaussian Mixture Model (GMM) representative of all speech data. We then generate speaker specific models based on their corresponding speech data. We upload these GMMs on the mobile phones of users and on the remote server. We also install HTK on all the mobile phones and the remote server. At run-time, a recorded audio file is converted to Perceptual Linear Predictive (PLP) coefficients file using the *HCopy* tool of HTK, and this PLP file is then compared with all the models of colocated users (determined through Bluetooth) using the *HERest* tool of HTK. Finally, the model with highest likelihood of match is assigned as the speaker model. The indoor localization and colocation detection is based on the *lightblue* [21] module for PyS60, which is a cross-platform Bluetooth API and provides a easy access to various Bluetooth operations. We use the accelerometer sensor API provided by PyS60 platform to access X, Y, Z axes data of accelerometer sensor.

[4]We also note that more fine-grained solutions based on Wi-Fi fingerprinting and coarse-grained solutions like logical localization [2] can be integrated into this module, but since our aim is to differentiate between work and social spaces, we limit this feature to the Bluetooth based technique.

5. MICRO-BENCHMARKS

We perform a two-part evaluation of the SociableSense system. Firstly, in this section, we evaluate the adaptive sampling scheme with respect to accuracy, energy, and latency, and computation distribution scheme with respect to selecting the best suitable configuration, based on real traces collected through participants carrying mobile phones. Secondly, in Section 6, we conduct a social psychology study to understand the usefulness of the system to social scientists and participants.

5.1 Sensor Sampling Benchmarks

5.1.1 Experimental Datasets

We collect the dataset for benchmarking the sensor sampling component in an office environment during day time working hours (10am to 4pm). We gather a total of 231 hours of raw accelerometer sensor data (i.e., X, Y, Z coordinates), 241 hours of Bluetooth data (i.e., Bluetooth identifiers), and 151 hours of microphone data (i.e., audio recordings), with 10 users carrying a Samsung Galaxy S or Nokia 6210 Navigator phone. The sampling of the accelerometer and Bluetooth sensors is performed continuously with a sleep interval of 0.5, 1 second, respectively. Audio samples of length equal to 5 seconds are recorded from the microphone sensor with a sleep interval of 1 second between consecutive recordings.

Samsung Galaxy S phones were running *Google Android version 2.1* [1] or higher. Android Bluetooth APIs were used to discover the Bluetooth devices in the proximity. The discovery process is asynchronous and the method call immediately returns with a boolean indicating whether discovery has successfully started. The discovery process involves an inquiry scan, followed by a page scan of each found device to retrieve its Bluetooth name. Android *SensorManager Service* is used to access X, Y, Z axes data of accelerometer sensor. This process is asynchronous too and involves registering a listener for capturing accelerometer data. Android SensorManager API provides various speeds at which data are sensed from accelerometer, and we use $SENSOR\_DELAY\_FASTEST$ setting to get sensor data as fast as possible. Nokia 6210 Navigator phones were running *Symbian S60*. We use the *lightblue* [21] module for PyS60 for performing Bluetooth discovery operations and the accelerometer sensor API provided by PyS60 platform to access X, Y, Z axes data of accelerometer sensor.

5.1.2 Methodology

Tuning of Adaptive Sampling Scheme We explore the performance of learning technique for each sensor for the entire parameter space of α (explained in Section 4.1) to fine-tune the adaptive sampling scheme. Since the dataset is collected in an office environment, the α value selected through this process should be general enough to provide the same level of performance in a similar environment.

Categorization of Events As discussed in Section 4.1, we classify the raw data from the sensor traces into events, which can be of two types, *viz.*, "unmissable" and "missable" events (or "interesting" and "uninteresting" events). In the case of the microphone sensor, an unmissable event corresponds to some audible data being heard in the environment and a missable event corresponds to silence. These events are generated using a GMM classifier [31] capable of classifying whether an audio trace contains any audible data. For the Bluetooth sensor traces, an unmissable event corresponds to a change in the number of colocated users, whereas a missable event indicates that there is no change. In the case of the accelerometer sensor, the unmissable event corresponds to movement of a user and a missable event indicates that the user is stationary. Although both these events are unmissable, it is sufficient to detect just one of them since we only have two possible events, so we choose a "user moving event" as unmissable.

Performance Metrics We evaluate the performance with respect to the metrics: *accuracy*, *energy*, and *latency*. The accuracy is measured in terms of the percentage of missed events. An event is said to be missed when there is an unmissable event recorded in the trace file while the sensor is not actively queried. The energy consumption is measured using the Nokia Energy Profiler (NEP) [13]. NEP is a stand-alone test and measurement application for Nokia phones, and it provides an easy way of measuring the power consumption of the mobile phone at a fine-grained time-intervals. The energy consumption of a task is computed as follows: we first measure the baseline energy consumption of mobile phone, and then we activate the task in SociableSense and measure the energy consumption. The difference between these energy consumption values is calculated as energy of the task. We repeat this procedure for many iterations and finally, we calculate the average of these values to determine the average energy consumption of the task. The latency is measured based on the delay in detecting change of event sequence from missable to unmissable and vice versa. For example, let us consider that a user is moving from time T_x to T_y, and the system detects this moving event at T_e ($T_x < T_e < T_y$). The latency of detecting this user moving event is calculated as $T_e - T_x$. All the performance metrics used are *the lower the better* type of metrics (as accuracy is measured in terms of missed events).

Techniques used for Comparison To quantify the advantages of using adaptive sensor sampling, we compare its performance with various static functions and a dynamic adaptation technique that controls the rate of sensor sampling based on the context of the user. We define two types of functions: if there are no "interesting" events observed (i.e., missable events), then the sampling interval increases (i.e., the sampling rate decreases) from its current value based on a *back-off function*. Similarly, if the event is classified as unmissable, then the sampling interval decreases (i.e., the sampling rate increases) from its current value based on an *advance function*. For example, let us assume that the current sleep interval between two consecutive audio samples is x; a categorization of missable and unmissable events can be as follows: missable: *silence*, unmissable: *some audible data*. If the classifier detects that the current event is silence, i.e., missable, then the sleep interval is increased to $f(x)$ based on a back-off function. If the choice of this function is quadratic then the resulting sleep interval will be $f(x) = x^2$. Similarly, if an unmissable event is detected, the sleep interval can be adjusted based on a quadratic advance function like $f(x) = \sqrt{x}$. The back-off and advance functions used for the evaluation are given in Table 1.

We also compare the learning scheme against the dynamic adaptation technique presented in [30]. This technique selects a sampling function dynamically according to the number of consecutive events of the same category (missable or

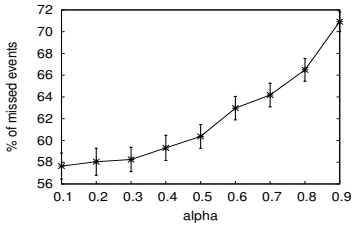


Figure 4: Accuracy (% of missed events) vs alpha for Bluetooth.

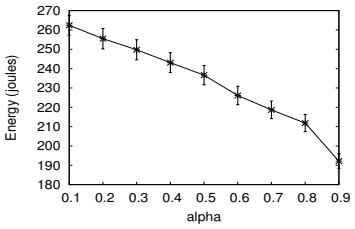


Figure 5: Energy consumption per hour vs alpha for Bluetooth.

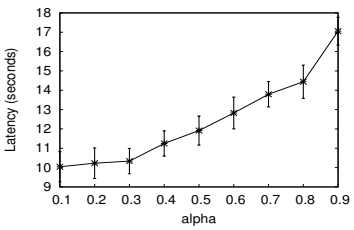


Figure 6: Latency vs alpha for Bluetooth.

Table 1: Back-off and advance functions

Type	Back-off function	Advance function
Linear	$k \times x$	x/k
Quadratic	x^2	$\sqrt{x}$
Exponential	e^x	$\log_e x$

unmissable). More specifically, given a set of sampling functions, the scheme initially uses the least aggressive sampling function (e.g., linear back-off if missable events are observed) and if the same type of events are detected for a number of times equal to a certain threshold then it switches to a more aggressive sampling function (e.g., quadratic back-off). This continues until it finally switches to the most aggressive sampling function (e.g., exponential back-off). The benchmarks for each of the sensors are evaluated for various combinations of advance and back-off functions, including the dynamic adaptation and learning techniques. The x-axis labels in this subsection have the format *advance–back-off functions*.

5.1.3 Results

To optimize the learning based technique for each of the sensors, we first measure its performance in terms of accuracy, energy, and latency by varying the α value. Figures 4, 5, and 6 show the variation of the parameter α with respect to the performance of the learning based technique for the Bluetooth sensor in terms of accuracy, energy, and latency, respectively. Based on these plots, we choose α to be 0.5 as this value corresponds to higher accuracy and lower latency without considerable compromise on energy. Since we measure the *sociability* with respect to colocation as well, accurately detecting it (using Bluetooth) is an important consideration. We also note that since the dataset is collected in an office environment, the α value selected may provide close performance in a similar type of workplace.

We then evaluate the performance of various combinations of advance and back-off functions with respect to all the performance metrics. The results for the Bluetooth sensor are shown in Figures 7, 8, and 9. We can observe that the learning based technique performs well with respect to accuracy and latency with only a slight increase in energy. The accuracy of the learning based technique is 12% more and latency is 56% less than that of the dynamic adaptation technique (*dynamic-dynamic*), while the energy consumption of the former is 25% more than the latter. Compared to the other dynamic combinations like *exponential-dynamic*, *quadratic-dynamic*, the performance of the learning based technique is very close with respect to energy and accuracy, however, the latency is as much as 40% lesser than these combinations. We can observe that the combinations *quadratic-linear* and *exponential-linear* that use linear back-off function exhibit similar performance, however, the gains of the learning based technique with respect to accuracy and latency are higher. Figures 10, 11, and 12 show the performance of all the techniques for the accelerometer sensor with respect to accuracy, energy, and latency, respectively. We can observe that the learning based technique performs much better than the other techniques in terms of accuracy and latency, and with respect to energy consumption, it has performance similar to other techniques (the maximum difference in energy consumption is only 4%). In other words, the learning based technique performs better than all the combinations of dynamic scheme with respect to all the performance metrics for accelerometer sensor. Therefore, these results justify the use of the learning based technique for the adaptive sampling of sensors. Following similar considerations, which we do not present here due to space limitations, we choose to use the learning method also for the microphone sensor.

5.2 Computation Distribution Benchmarks

In this section, we present the evaluation of the computation distribution scheme through micro-benchmark tests.

5.2.1 Methodology

The main aim of these benchmarks is to evaluate the performance of the utility function (explained in Section 4.2) used in the computation distribution scheme in terms of selecting the best configuration given the importance assigned to each performance dimension. More specifically, we compare the performance of all possible configurations and evaluate whether the utility function selects the appropriate configuration based on the weights given for the three dimensions: energy, latency, and total data sent over the network. The performance metrics used are: *lifetime of the phone* (i.e., the total time until battery gets completely discharged from the fully charged state), *average latency* (i.e., the average time taken for processing a sensor task), and *average data sent over the network* (i.e., the average number of bytes sent by the system over the 3G network to process a sensor task).

5.2.2 Tasks used in the Benchmarks

The classification tasks used in the benchmarks are of three types based on the sensor from which data are queried. The *activity recognition* classification task has one subtask that classifies the data sensed from the accelerometer sensor into *moving* or *idle* states. The *colocation detection* classification task has one subtask that detects the change in colocation of user observed through the change in colocated Bluetooth devices. The *speaker identification* classification task performs the speaker identification and is divided into

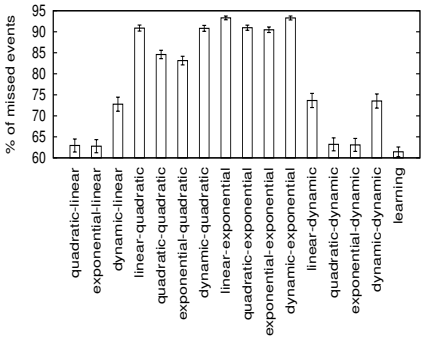


Figure 7: Accuracy of the adaptation techniques for Bluetooth.

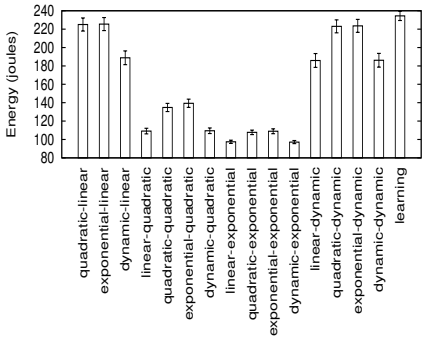


Figure 8: Energy consumption per hour of the schemes.

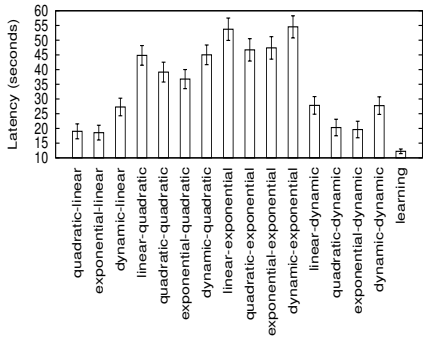


Figure 9: Latency of the adaptation techniques for Bluetooth.

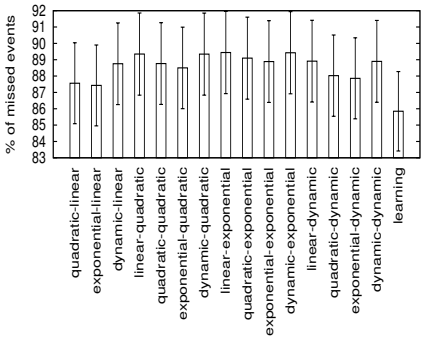


Figure 10: Accuracy of the adaptation schemes for accelerometer.

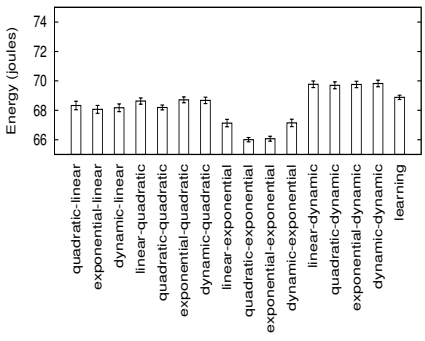


Figure 11: Energy consumed per hour of the adaptation techniques.

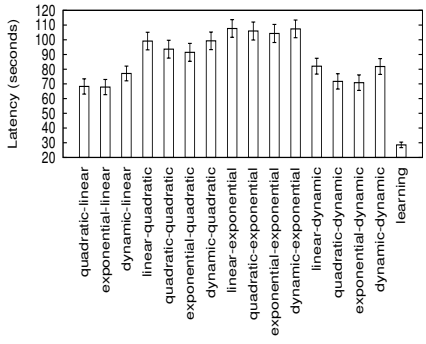


Figure 12: Latency of the adaptation techniques for accelerometer.

two subtasks: the first subtask converts the recorded audio sample to Perceptual Linear Predictive (PLP) coefficients file using the *HCopy* tool of the Hidden Markov Model Toolkit (HTK) [17], and the second subtask compares the extracted coefficients file with the speaker models of all users using the *HERest* tool of HTK, and the model with the highest likelihood of match is selected as the speaker model of the recorded audio file.

5.2.3 Results

We first present the selection of the best configuration by the computation distribution scheme for the speaker identification classification task. This task consists of two subtasks, each of which can be executed locally on the phone or remotely in the cloud. Therefore, we have a total of four configurations in which this task can be executed: *C1:* all subtasks computed locally; *C2:* first subtask computed locally and the other remotely; *C3:* first subtask computed remotely and the other locally; *C4:* all subtasks computed remotely. Let *S1*, *S2*, *S3* and *S4* denote the various combinations of weights (configured by the experiment designers) in the utility function: *S1*: $w_e = 1, w_l = 0, w_d = 0$ (i.e., maximum weight to energy); *S2*: $w_e = 0, w_l = 1, w_d = 0$ (i.e., maximum weight to latency); *S3*: $w_e = 0, w_l = 0, w_d = 1$ (i.e., maximum weight to data sent over the network); *S4*: $w_e = 0.33, w_l = 0.33, w_d = 0.33$ (i.e., equal weights).

Figures 13, 14 and 15 show the performance of all the configurations with respect to energy consumption, latency, and total data sent over the network for processing the speaker identification task. We can observe that the utility function for the combination *S1* selects the configuration *C4* as this has the lowest energy consumption, and for the combination *S2* the configuration selected is *C4* as this is the lowest in terms of latency as well. For the combination *S3*, the configuration *C1* is selected as this sends the least amount of data over the network. Finally, for the combination *S4*, which gives equal weights to all the dimensions, the configuration *C4* is selected as this is the best considering all dimensions; moreover, it is also the best performing configuration in two out of three dimensions.

The above evaluation shows that the proposed scheme selects the best configuration for a given task given the weights defined by the experiment designers. However, to study the impact of this selection "at system level", we have also evaluated the effect of these decisions "at task level" on the overall performance of the system in terms of lifetime of mobile phone, average latency of processing a task, and average data sent over the network for processing a task. The phone battery capacity is 3.7v/750mAh. We consider three sensor classifiers: activity recognition (one subtask), change in colocation detection (one subtask), and speaker identification (two subtasks), so we have in total four possible subtasks each of which can be computed locally on the phone or remotely in the cloud. At a system level, this results in 16 possible ways (or configurations *C1* to *C16*) of processing the sensor tasks. We evaluate the utility function for the same combination of weights: *S1*, *S2*, *S3*, and *S4*. We use the same sensor traces used in the evaluation of the sensor sampling scheme, and the learning based technique is used as sampling mechanism.

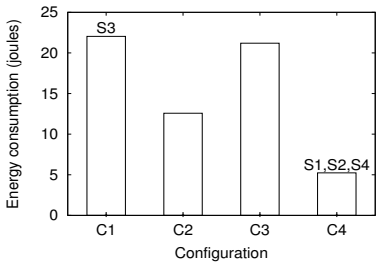


Figure 13: Energy consumption for processing the speaker identification task.

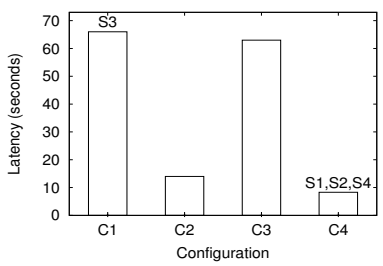


Figure 14: Latency or delay for processing the speaker identification task.

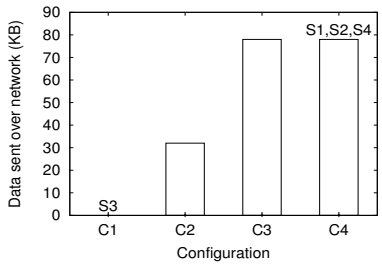


Figure 15: Data sent over the network for processing the speaker identification task.

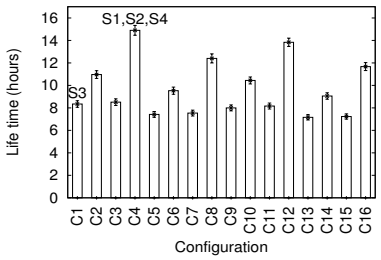


Figure 16: Total lifetime of the mobile phone.

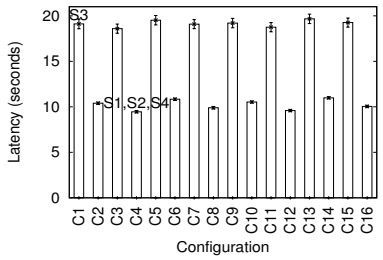


Figure 17: Average latency of processing a task.

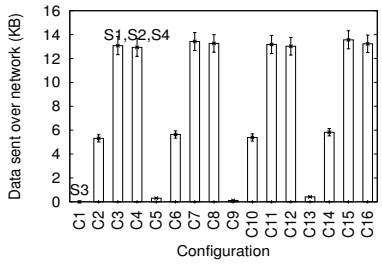


Figure 18: Average data sent over the network for processing a task.

Figures 16, 17 and 18 show the performance of all the possible configurations in terms of lifetime, latency, and total data sent over the network. We can observe that for the combination *S1* that gives maximum weight to energy, the configuration with the highest overall lifetime is selected (i.e., *C4*) . For the combination *S2* that gives maximum weight to latency, the configuration *C4* is selected as this is the lowest in terms of latency as well. For the combination *S3*, the configuration with lowest data sent over the network is selected (i.e., *C1*). We can also observe that there are multiple configurations with the lowest amount of data sent over the network like *C1*, *C5*, *C9*, *C13* and among them the configuration with better latency and lifetime values (i.e., *C1*) is selected. Finally, for the combination *S4*, the configuration *C4* is selected as it is much better in two out of three dimensions compared to the other configurations. These results show that the proposed computation distribution scheme selects the best configuration given the weights assigned to the different dimensions.

6. SOCIAL PSYCHOLOGY STUDY

In addition to the micro-benchmark tests, we also conduct a social study to evaluate the usefulness of the SociableSense system to the social scientists and the participants.

6.1 Overview of the Experiment

We conduct the experiment for a duration of two working weeks (10 days) involving 10 users. We divide the experiment into two phases each of which lasts for a week. In the first phase, the feedback mechanisms of the SociableSense system were disabled, and in the second phase they were displayed. More specifically, we show the following to users: sociability, strength of their relations, activity levels, and alerts about the users in sociable locations. During the experiment each user carries a Nokia 6210 Navigator mobile phone. We identify five main locations where users generally spend most of their time. We categorize three of these as *work locations* as users work spaces are located in these locations, and two of the locations as *sociable locations* that include a common room and a cafeteria where users either socialize or have breakfast/lunch. These two categories of locations are in different parts of the building, i.e., Bluetooth ranges were non-overlapping, therefore, localization error is negligible.

6.2 Results and Discussion

We study the effect of feedback mechanisms on the sociability of the users. Figures 19 and 20 show the *colocation network constraint* and *interaction network constraint* for the two phases of the experiment. We can observe that the average network constraint for both colocation and speech networks is lower when feedback and alert mechanisms were enabled. Note that the *network constraint* is a *lower the better* type of metric for sociability. We can also observe that the difference between the network constraints with and without feedback is greater in sociable locations than in work locations, and based on this we can infer that the feedback mechanisms have a greater effect on individuals at sociable locations. We conjecture that this is the case because of the opportunities of interacting in these locations.

Figure 21 shows the standard deviation of network constraint of all the participants of the experiment. The standard deviation is similar for both the constraints with and without feedback mechanisms except for the sociable locations. This shows that in the sociable locations the difference in the level of sociability among participants is lesser with feedback mechanisms. From this we can infer that in sociable locations with feedback mechanisms most users participate in interactions compared to the case without feedback mechanisms. Figure 22 shows the average network con-

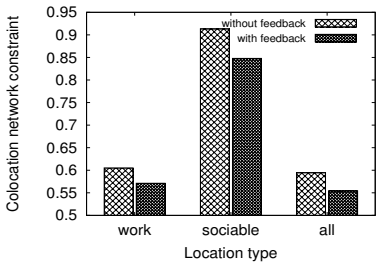


Figure 19: Colocation network constraint vs location types.

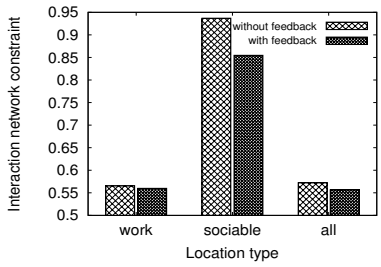


Figure 20: Interaction network constraint vs location types.

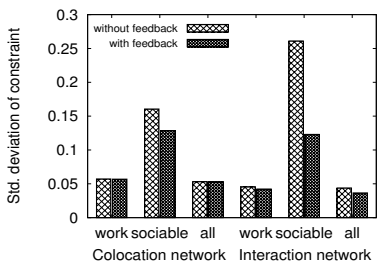


Figure 21: Standard deviation of network constraint.

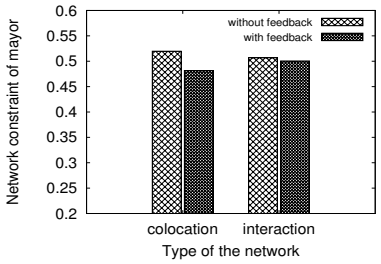


Figure 22: Average network constraint of the mayor.

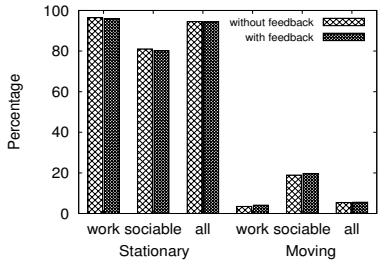


Figure 23: Activity levels of users in various location types.

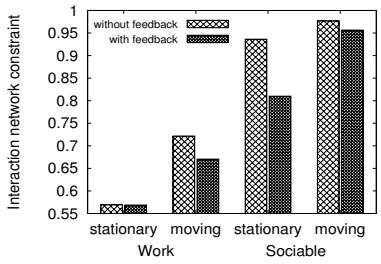


Figure 24: Network constraint vs activity levels.

straint of the mayor of the group, and we can observe that the level of sociability achieved by the mayor is very close in both the phases of the experiment. This maybe related to the fact that the feedback mechanisms do not modify the behavior of a person that is already very sociable.

Finally, we analyze the effect of feedback mechanisms on the level of activity of the users. Figure 23 shows the level of activity of users in both the phases of the experiment, where we observe a fairly consistent behavior. In order to understand the effect of activity level on sociability of users, we also measure the correlation between the level of activity and interaction network constraint, which is shown in Figure 24. Apart from the observation that feedback mechanisms help users to socialize more irrespective of their level of activity, this result also shows another interesting phenomenon: in the work locations the feedback mechanisms have greater effect on the sociability of users when they are physically active, whereas in sociable locations the feedback mechanisms are effective when users are stationary.

For this study we give equal importance to all the dimensions: energy, latency, and data sent over the network. The combination of weights is same as that of *S4* in Section 5.2. Therefore, the computation distribution scheme selects the configuration *C4* (speaker identification task computed remotely and other classification tasks computed locally). The use of computation distribution scheme leads to approximately 28% more battery life, 6% less latency per task, and 3% less data transmitted over the network per task compared to the model where all the classification tasks are computed remotely.

7. RELATED WORK

Recently, ubiquitous technologies and especially mobile smart phones have been widely used to monitor user behavior [16, 24]. In particular, EmotionSense [31] is a platform that is able to detect user emotional states using standard off-the-shelf phones. Experience sampling [16] is a widely used technique for social psychological studies, however, it requires constant user input and is found to be generally biased [28]. Other examples of systems that exploit ubiquitous technology to quantitatively measure human behavior include the Sociometer [8], the Mobile Sensing Platform (MSP) [7]. Several studies were performed using the Sociometer in order to understand organizational behavior [32]. These systems rely on purpose-built devices (such as mote-class sensors) that are not part of the everyday life of people and, therefore, can be felt as obtrusive or simply may not be always carried by the users of an experiment.

Energy is a key issue in building mobile sensing systems. For example, the EEMSS platform [33] uses a hierarchical sensor management strategy to recognize participants' activities. The Jigsaw continuous sensing engine [22] balances the performance requirements of the applications and the resource demands of sensing continuously on the mobile phone. Llama [4] is an energy management system based on the user statistics in terms of usage and recharge cycles and exploits the excessive energy for a better user experience. PRISM [11] is a mobile sensing platform that makes it easier to develop and deploy the *participatory applications* as executables on the mobile phones of the participating users. Virtual Compass [3] builds a relative neighbor graph using radio technologies like Bluetooth and Wi-Fi, and can be a useful tool for social applications. It achieves energy efficiency by monitoring topology changes and adapting scanning rates, and selecting the most appropriate radio interface based on energy characteristics.

With respect to exploiting the cloud computation, in [25] the authors show that continuous sensing is a viable option for mobile phones by adopting strategies for efficient data uploading. The focus of this work is on the minimization of the amount of data sent to back-end servers and not on the distribution of the computation. CloneCloud [9]

enables mobile applications running in an application-level virtual machine to offload part of the execution to device clones running in the cloud. However, the system is mainly useful for mobile applications running in a virtual machine. Finally, the authors of [10] present MAUI, a system that achieves energy efficiency through fine-grained code offloading to the cloud, however, it requires developers to annotate the application source code. With respect to these works, the computation distribution component in our system distributes tasks among local and cloud resources considering the requirements of the experiment designers in terms of the battery consumption, delay, and traffic trade-offs. To the best of our knowledge, this is the first mobile sensing system that implements energy-efficient mechanisms that are adaptive with respect to the changing context and optimized according to the requirements of the researchers collecting the sensor data. Moreover, unlike SociableSense, all these existing systems are used for collecting data without providing any feedback to the users.

8. CONCLUSIONS

In this paper, we have presented SociableSense, a sensing platform that implements a novel adaptive sampling scheme based on learning methods, and a dynamic computation distribution mechanism based on decision theory. The tool, based on smart phones, has been applied to quantitatively measure the sociability of users and strength of their relations with each of their colleagues in workplaces. It *closes the loop* by providing real-time feedback to users about their *sociability*, the strength of their relations with colleagues, and opportunities to interact.

The sensor sampling scheme adaptively samples from accelerometer, Bluetooth, and microphone sensors while balancing energy-accuracy-latency trade-offs. The computation distribution scheme distributes the classification tasks between local and remote resources while considering the importance given by the experimenter to each of the dimensions: energy, latency, and data traffic. We have evaluated the performance of sensor sampling and computation distribution schemes using several micro-benchmark tests. By means of a social study we have also demonstrated the usefulness of SociableSense to researchers and participants.

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 psychology experiment. This work was supported through Gates Cambridge Trust, and EPSRC grants EP/I019308, EP/G069557, and EP/I032673.

9. REFERENCES

[1] Android 2.1. http://developer.android.com/sdk/android-2.1.html.

[2] M. Azizyan, I. Constandache, and R. Roy Choudhury. SurroundSense: Mobile Phone Localization via Ambience Fingerprinting. In *Proc. of MobiCom'09*. ACM, 2009.

[3] N. Banerjee, S. Agarwal, P. Bahl, R. Chandra, A. Wolman, and M. Corner. Virtual Compass: Relative Positioning to Sense Mobile Social Interactions. In *Proc. of Pervasive'10*. LCNS Springer, 2010.

[4] N. Banerjee, A. Rahmati, M. Corner, S. Rollins, and L. Zhong. Users and Batteries: Interactions and Adaptive Energy Management in Mobile Systems. In *Proc. of UbiComp'07*. ACM, 2007.

[5] G. H. Bower and E. R. Hilgard. *Theories of Learning*. Prentice-Hall, Inc., 1975.

[6] R. Burt. *Structural Holes: The Social Structure of Competition*. Harvard University Press, 1995.

[7] 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, and D. Wyatt. The Mobile Sensing Platform: An Embedded Activity Recognition System. *IEEE Pervasive Computing*, 7(2):32–41, 2008.

[8] T. Choudhury and A. Pentland. Sensing and Modeling Human Networks using the Sociometer. In *Proc. of ISWC'03*, 2003.

[9] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti. CloneCloud: Elastic Execution between Mobile Device and Cloud. In *Proc. of EuroSys'11*. ACM, 2011.

[10] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl. MAUI: Making Smartphones Last Longer with Code Offload. In *Proc. of MobiSys'10*. ACM, 2010.

[11] T. Das, P. Mohan, V. N. Padmanabhan, R. Ramjee, and A. Sharma. PRISM: Platform for Remote Sensing using Smartphones. In *Proc. of MobiSys'10*. ACM, 2010.

[12] D. Donoho. Compressed Sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, 2006.

[13] Nokia Energy Profiler. http://store.ovi.com/content/17374.

[14] P. C. Fishburn. Independence in Utility Theory with Whole Product Sets. *Operations Research*, 13(1):28–45, 1965.

[15] P. C. Fishburn. Utility Theory. *Management Science*, 14, 1968.

[16] J. Froehlich, M. Y. Chen, S. Consolvo, B. Harrison, and J. A. Landay. MyExperience: A System for In situ Tracing and Capturing of User Feedback on Mobile Phones. In *Proc. of MobiSys'07*. ACM, 2007.

[17] Hidden Markov Model Toolkit. http://htk.eng.cam.ac.uk.

[18] L. Kaelbling, M. Littman, and A. Moore. Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[19] R. Keeney and H. Raiffa. *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*. John Wiley & Sons, 1976.

[20] R. L. Keeney. Common Mistakes in Making Value Trade-Offs. *Operations Research*, 50(6):935–945, 2002.

[21] lightblue. http://lightblue.sourceforge.net.

[22] H. Lu, J. Yang, Z. Liu, N. Lane, T. Choudhury, and A. Campbell. The Jigsaw Continuous Sensing Engine for Mobile Phone Applications. In *Proc. of SenSys'10*. ACM, 2010.

[23] W. Lynn, B. N. Waber, S. Aral, E. Brynjolfsson, and A. Pentland. Mining Face-to-Face Interaction Networks using Sociometric Badges: Predicting Productivity in an IT Configuration Task. In *Proc. of the ICIS'08*, 2008.

[24] E. Miluzzo, C. T. Cornelius, A. Ramaswamy, T. Choudhury, Z. Liu, and A. T. Campbell. Darwin Phones: The Evolution of Sensing and Inference on Mobile Phones. In *Proc. of MobiSys'10*. ACM, 2010.

[25] M. Musolesi, M. Piraccini, K. Fodor, A. Corradi, and A. T. Campbell. Supporting Energy-Efficient Uploading Strategies for Continuous Sensing Applications on Mobile Phones. In *Proc. of Pervasive'10*. LCNS Springer, 2010.

[26] K. S. Narendra and M. A. L. Thathachar. *Learning Automata: An Introduction*. Prentice-Hall, Inc., 1989.

[27] D. Olguin and A. Pentland. Assessing Group Performance from Collective Behavior. In *Proc. of the CSCW'10*. ACM, 2010.

[28] D. L. Paulhus and D. B. Reid. Enhancement and Denial in Socially Desirable Responding. *Journal of Personality and Social Psychology*, 60(2):307–317, 1991.

[29] Python for S60. https://garage.maemo.org/projects/pys60.

[30] K. K. Rachuri, M. Musolesi, and C. Mascolo. Energy-Accuracy Trade-offs in Querying Sensor Data for Continuous Sensing Mobile Systems. In *Proc. of Mobile Context-Awareness Workshop'10*, 2010.

[31] K. K. Rachuri, M. Musolesi, C. Mascolo, P. J. Rentfrow, C. Longworth, and A. Aucinas. EmotionSense: A Mobile Phones based Adaptive Platform for Experimental Social Psychology Research. In *Proc. of UbiComp'10*. ACM, 2010.

[32] B. Waber, D. O. Olguin, T. Kim, and A. Pentland. Productivity through Coffee Breaks: Changing Social Networks by Changing Break Structure. In *Proc. of SSNC'10*, 2010.

[33] Y. Wang, J. Lin, M. Annavaram, Q. A. Jacobson, J. Hong, B. Krishnamachari, and N. Sadeh. A Framework of Energy Efficient Mobile Sensing for Automatic User State Recognition. In *Proc. of MobiSys'09*. ACM, 2009.