Smartphones based Social Sensing: Adaptive Sampling, Sensing and Computation Offloading

Kiran K. Rachuri

University of Cambridge
Computer Laboratory
St. John's College

2012

This dissertation is submitted for
the degree of Doctor of Philosophy

Declaration

This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration except where specifically indicated in the text.

This dissertation does not exceed the regulation length of 60 000 words, including tables and footnotes.

Smartphones based Social Sensing: Adaptive Sampling, Sensing and Computation Offloading

Kiran K. Rachuri

Summary

In recent years the number of smartphone users has been increasing at an unprecedented rate. Smartphones are carried by billions of people every day, and are equipped with many sensors such as accelerometer, microphone, and GPS. These sensors can be used to capture various social and behavioural aspects of the users, which we refer to as social sensing. An important application area of social sensing is the support to experimental social psychology. However, smartphones have limited battery and processing power, which pose challenges to the capture of data from the sensors and to data processing. In this dissertation we present techniques to efficiently capture and process data from the smartphone sensors and show that smartphones can be effective tools to perform social sensing and to conduct social psychological studies.

The sensors embedded in the phone have to be sampled often in order to capture the user's behaviour. This, however, may lead to faster depletion of the battery. If the sensors are sampled at a slower rate, then it may not be possible to accurately capture the user's behaviour. To meet the challenges posed by phone sensing, we design three adaptive schemes. First, we design an adaptive sampling framework that samples the data from the sensors considering the user's context to conserve energy, while providing the required accuracy to the applications. Second, to further increase the energy efficiency of capturing data, we design a framework that exploits the sensors in buildings and dynamically distributes the sensing tasks between the local phone and the infrastructure sensors. When the data from the sensors are captured, they need to be classified to derive high-level inferences. Third, to efficiently process the data we design a computation offloading scheme that decides whether to compute the classification tasks locally on the phone or remotely in the cloud by considering various dimensions such as energy, latency, and data traffic.

We then demonstrate the ability of smartphones to perform social sensing and to conduct social psychological studies by designing and deploying three applications that use the services of the proposed schemes. First, we present EmotionSense, a passive monitoring application that captures the user's emotions and speech patterns automatically by using the sensors in off-the-shelf mobile phones. Second, we present WorkSense, a workplace behavioural monitoring application that can detect the collaboration and interaction patterns of the workers. Third, we present SociableSense, a persuasive and behavioural monitoring application that aims to foster interaction and relations between the users at the workplace. We also report on the deployment of these applications in real environments.

Acknowledgments

First of all, I am grateful to my supervisor Cecilia Mascolo for her support, encouragement, and guidance throughout my Ph.D. Her feedback on my research, papers, and dissertation has been invaluable. I also thank her for opening up outstanding collaborations, which culminated in fruitful results. I would like to thank my collaborators Mirco Musolesi, Christos Efstratiou, Ilias Leontiadis, Neal Lathia, Theus Hossman, and Andrius Aucinas. The learning I had from these collaborations is priceless. Special thanks to Peter Jason Rentfrow for collaborating with us and helping us understand the field of social sciences. I learnt a lot about social sciences, its research methodology, existing work, and designing social experiments by collaborating with Jason.

I am grateful to Andy Hopper and Jean Bacon for their comments and feedback on my research work. I thank the members of Network and Operating Systems group, Computer Laboratory for their feedback on my papers. I made many friends in the lab, Amitabha, Bence, Chloe, Daniele, Enzo, Harris, Jisun, John, Kharsim, Liam, Narseo, Nishant, Salvo, Sarfraz, Tassos, indeed too many to name here. I thank them for many lively and interesting conversations. I thank Piete Brookes, Lise Gough, Carol Nightingale, Tanya Hall for their help in various administrative matters.

My sincere thanks to St. John's college for their support during my Ph.D. I am indebted to the Gates Cambridge Trust for awarding me a generous scholarship. I also thank the Gates Cambridge Trust, the Faculty of Computer Science and Technology, and St. John's College for providing me travel grants to present papers at the UbiComp 2010, MobiCom 2011, and PerCom 2013 conferences.

I would like to acknowledge my friends, Hareesh, Arpita, Srinath, Deepa, Harish, Swaroopa for making my Ph.D. journey lively and fun. I deeply thank each of my family members for their support and encouragement. I am indebted to my parents for their unconditional love, support, and sacrifices. My wife Lavanya has been a source of constant encouragement and support throughout my Ph.D. and M.S., and this dissertation would not have been possible without her love and support.

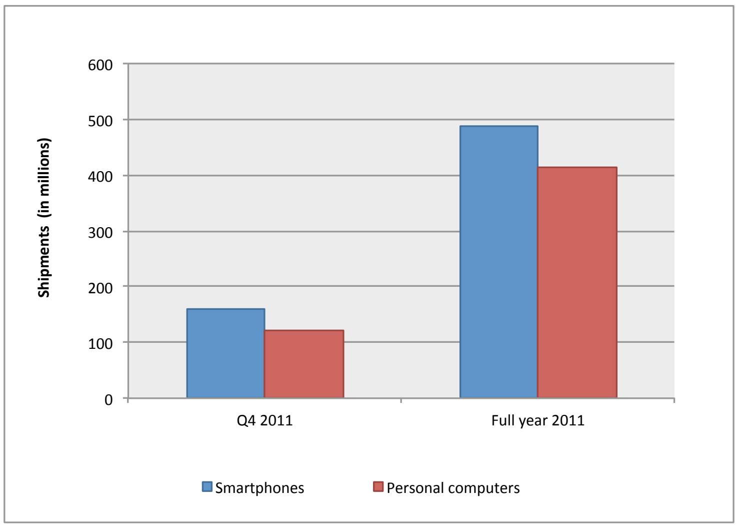
To my parents and my wife

Contents

1 Introduction

1.1 Smartphones

Until a few years ago mobile phones were mainly used as communication devices for phone calls and Short Message Service (SMS) messages. They typically had a screen, keyboard, and Global System for Mobile Communications (GSM) radio, and most of the emphasis was on calls and messages. All this seems to have changed with the advent of phones like the Nokia N95, Blackberry, and Apple iPhone, some of the first consumer-oriented smartphones. The Oxford English Dictionary defines a smartphone as “a mobile phone that is able to perform many of the functions of a computer, typically having a relatively large screen and an operating system capable of running general-purpose applications”. A smartphone is a mobile phone but has many more advanced capabilities like powerful CPUs, large memory, sensors like accelerometer, proximity, Global Positioning System (GPS) etc., and advanced connectivity options like Bluetooth, Wi-Fi, and mobile data network. The first generation Apple iPhone, for example, included many advanced connectivity features and sensors such as EDGE (Enhanced Data rates for GSM Evolution) and Wi-Fi connectivity, accelerometer, proximity, and ambient light sensors. The sensors were included to enhance user experience (for example, rotating the screen by detecting when the user changes the position of the phone using the accelerometer sensor) and to extend battery life (for example, automatic adjustment of brightness using the light sensor). Today, there are many smartphone vendors like Google, Microsoft, Research In Motion (RIM), Sony, and Samsung, and there are many smartphone operating systems



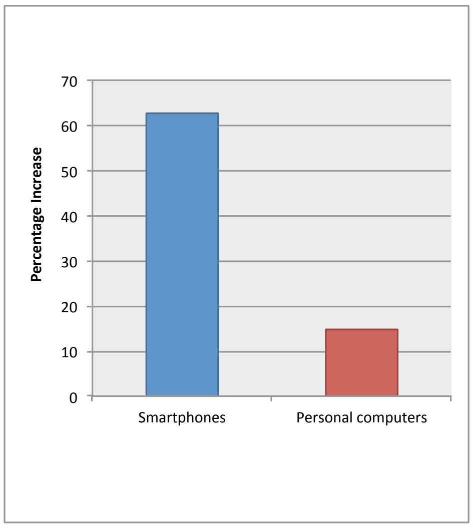


Figure 1.1: Smartphone and personal computer shipment statistics. (Source: Canalys 2012)

Figure 1.2: Growth rates 2011/10. (Source: Canalys 2012)

available including Apple iOS, Google Android, RIM Blackberry OS, Windows Mobile, and Nokia Symbian OS. The availability of so many options has led to the unprecedented growth of the smartphone market around the world.

The number of smartphone users has been growing exponentially so that they now have surpassed the number of personal computers: vendors shipped 488 million smartphones and 415 million personal computers in 2011 [CAN11]. Further, smartphone shipments in 2011 increased by 62% over those in 2010, while personal computer shipments increased only by 15% in 2011 above the 2010 shipments (Figures 1.1 and 1.2). This is a clear indication of a dramatic shift in the way people have been using computing devices. Furthermore, with the advent of application stores such as Apple App Store [1], Google Play [2], and Windows Marketplace [3], it is easy for users to explore and download applications. For example, it was reported in [COS12] that 82% of the time people spent on mobile media is via mobile applications. Due to their many advanced features and utilities smartphones have become part of the everyday life of billions of people.

The proliferation of smartphones has given a push to their use in many domains, spanning a variety of fields. Examples include navigation systems like maps [GMP], entertainment systems like gaming [ZCCM12], social networking services like Facebook [4] and Twitter [5], product recommendation systems [vRGMF09], social psychological systems [FCC$^+$07], and human mobility prediction systems [DGP12]. Further, deploying applications in

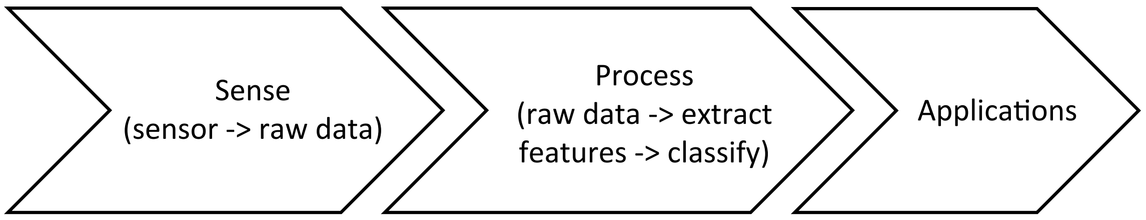
[1]http://www.apple.com/uk/iphone/from-the-app-store/
[2]https://play.google.com/store
[3]http://www.windowsphone.com/en-US/marketplace
[4]http://www.facebook.com/
[5]https://twitter.com/

the mobile user market has become easier for the application providers due to online application stores: once an application is ready it is only a few clicks away from being deployed to all the users' phones through these markets.

In the last few years the rapid adoption of smartphones, combined with their sensing, processing, and communication capabilities has also attracted considerable attention from the research community. Researchers have shown the potential of smartphones through many innovative systems to: track the physical activity of users [PPC+12], provide real-time trip information to passengers with the expected fare and trip duration [BNJ11], share and query global information [GLC+08], automatically tag digital images [QBRCN11] by sensing people and context, and enhance the experience of users in multiplayer games [ZCCM12].

1.2 Smartphone Sensing

Smartphone sensing is the process of capturing data from the sensors embedded in the phone. There are many sensors in modern smartphones such as the Samsung Galaxy SII or the Apple iPhone 4S: accelerometer, compass, GPS, microphone, proximity, to name a few. If we also consider the radios in the phone to be sensors (as they can also be used to capture the user's data), then the list increases: Bluetooth, GSM, Wi-Fi, NFC (Near Field Communication). The data captured from smartphones can be processed to draw inferences about the user such as about physical activity, interaction, and location. This typically involves extracting characterising features from the data and then classification to draw inferences. The flow of the process is as shown in the following figure.



The application of smartphones in many fields is enhanced by the addition of sensors. Researchers have devised innovative ways of using these sensors, such as location-based reminders [LFR+06], automatically updating the user's social network status with his/her current activity [MLF+08], fall detection for the elderly [YKE+10], and enhancing user experience in games [ZCCM12]. The CenceMe system [MLF+08] uses the accelerometer sensor in the mobile phone for activity recognition, and the microphone sensor to detect conversations. Many phone sensing research systems have also been proposed recently [MLF+08, GLC+08, LBBP+11, ACRC09].

1.3 Smartphone Sensing for Social Psychology

Smartphones can be used to capture a variety of behaviour and social aspects of the user: the microphone can be used to detect whether the user is speaking [LBBP^{+}11], the Bluetooth radio can be used to determine the users in proximity and thereby detect co-location, and GPS to detect the location of the user. By combining GPS co-ordinates with data from other sensors, the location can be further classified into categories like cafe, restaurant etc. [CLL^{+}12]. We refer to the sensing of various behavioural and social aspects of users through smartphones as ***Social Sensing***. One of the important research areas in which smartphone social sensing can be applied in is the social psychological sciences. In this dissertation we explore phone sensing in the conduct of social psychological research.

Social psychological sciences deal with the study of general behavioural and interaction patterns of users. They involve studying many aspects of users, including workplace behaviour, behavioural differences in a variety of locations (home, work etc.) and across social groups, intervention and feedback mechanisms and their effectiveness, and emotional patterns. Generations of social psychologists have tried to answer the following questions using methods such as direct observation [KSFS05], self-reporting [Tou99], and experience sampling [FCC^{+}07] involving social experiments on human subjects.

- **Interactions and emotions.** *How does location affect interaction? Do people interact more at home or at work? What emotions are typically exhibited? How does the frequency of interaction patterns and emotions vary? Can we measure emotions quantitatively? What is the correlation of emotion with location and activity or with interaction? How do speech patterns in a group of users vary over a given period?*

- **Workplace.** *What are the interaction and co-location patterns of the users in office or corporate environments? Do people interact more in personal office spaces or in common spaces like coffee rooms? Which workgroup members socialise with one another and why? How do collaboration patterns of workgroups evolve? Do these collaboration patterns differ significantly from those specified in project plans?*

- **Relations and feedback.** *Are users aware of their relations with their colleagues/-family/friends? And do they make an effort to revive social links? Is the behaviour of users significantly different at home and at work? What is the effect of colocation on these interaction patterns? If alerted to the risk of fading relations (e.g., with someone they have not talked to in a while), will they respond to that? Who attains social status in family/friend/work groups? What role does positive and negative affect play in interactions?*

The current research methods used to answer these questions make little use of technology. In addition to traditional self-reports, social scientists may also rely on one-time behavioural observation of the participants in laboratory settings [KSFS05]. Such methods may be useful, but the fact that they are based on behaviour in a lab raises concerns about their generalisability to non-lab contexts. Recently researchers have begun to use new methods in an effort to examine behaviour in everyday life. Daily diary [BDR03] and experience sampling methods [FCC+07, BB01, FMPP07], for example, ask participants to report the social events and psychological states they experienced either at the end of the day or periodically during the day. Another method [MP01] has used devices that take audio recordings (or snapshots) of participants' daily lives every few minutes, which are later transcribed and coded by teams of researchers. These methods have advantages over the traditional survey methods, but they are nevertheless not free from problems associated with forgetting events that took place during the day, and carrying an additional obtrusive electronic device.

Currently, social psychology studies do not use modern sensing technology to its full potential. Cameras and microphones have been used in the past, but studies have mainly been performed through direct observation [KSFS05] and questionnaires [BDR03], at the expense of the researcher's time. Most research in the social sciences, and social psychology in particular, relies almost entirely on self-reporting methods [MP01] and one-time behavioural observation of individuals in a laboratory. In social studies "at large", participants are either asked every few hours to complete a questionnaire on their location, activity, and social interactions, or, in more technology-supported studies, they carry a personal digital assistant that takes audio recordings every 20 minutes. These are then coded on several dimensions by teams of researchers. Such methods are better than one-time assessments, but they can be intrusive and labour-intensive, and are also found to be biased towards pleasant experiences [FMPP07, PR91].

A common class of biases concerns social desirability, or the tendency of people to respond to survey items in ways that present them in a favourable light. People may also engage in socially desirable responding because they lack sufficient insight or knowledge to accurately respond to survey items [PR91]. Even if they are performed *in situ* (as in the case of MyExperience [FCC+07]), self-reports, as they are used in daily diary studies, are especially prone to errors of memory. Indeed, several studies have shown that retrospective reports of thoughts, feelings and behaviour are unreliable and biased [FMPP07]. For instance, there is evidence that people behave consistently over a period of time and over a variety of situations [MGP06], that individuals exhibit mood and emotions similar to those of people with whom they interact most frequently [ZAT+05], and that most social activity of individuals is effectively neutral [CW88].

Smartphone sensing technology is capable of bringing a new perspective to the design of social psychology experiments, both in terms of accuracy of results of the social studies and from a practical point of view. There are several advantages of using mobile phones in conducting social studies.

- **Ubiquity.** Smartphones are already carried by billions of people across the world. It was reported [ENG12] that over 50% of US mobile users own smartphones. More importantly, they are an integral part of their everyday lives and a considerable amount of time is spent interacting through them. For example, it was reported in [DBB11] that Americans spend 2.7 hours per day socialising on their mobile phone.

- **Unobtrusiveness.** Smartphones are unobtrusive unlike purpose-built devices that must be carried in the experiments and are generally a burden to the participants. Smartphones are already carried by users, and their presence is likely to be "forgotten" by them, leading to accurate observation of spontaneous behaviour. This is especially true when the sensing is performed passively, without requiring any input from the user.

- **Sensor-richness.** Due to the presence of many sensors in mobile phones such as accelerometer (can be used for activity recognition), microphone (speaker, conversation detection), and GPS (location), the behaviour of users can be captured accurately and automatically.

- **Powerful processors.** Mobile phones are equipped with powerful processors, for example, the Samsung Galaxy SIII is equipped with a *Quad-core 1.4 GHz Cortex-A9* processor. It is feasible, therefore, to perform powerful classification and inference tasks locally on the phone without using the user's data plan to transmit the data elsewhere for processing.

- **Cloud connectivity.** Even though modern mobile phones are equipped with powerful processors, some classification tasks [KAH$^+$12, CBC$^+$10] may require high-end processing power that is available in data-centres or cloud farms. This type of task can be performed by transmitting the data to the cloud using high-speed connectivity options like Wi-Fi and 3G/4G available on mobile phones. Further, the accuracy of some classification tasks such as speech recognition (e.g., Siri on the iPhone 4S[6]) can be improved by using the "dictionaries" or other data available in the remote servers.

- **Ease of deployment.** It is easy to deploy social applications on smartphones, given the availability of online application stores. More importantly, since the user base of some of these systems is very large[7], there is an opportunity to conduct experiments with a large number of users, a distant possibility a few years back.

[6]http://www.apple.com/iphone/features/siri.html

[7]It was reported in the Google I/O developer conference 2012 that 400 million Android devices have now been activated and one million new Android devices are activated each day [BGR12].

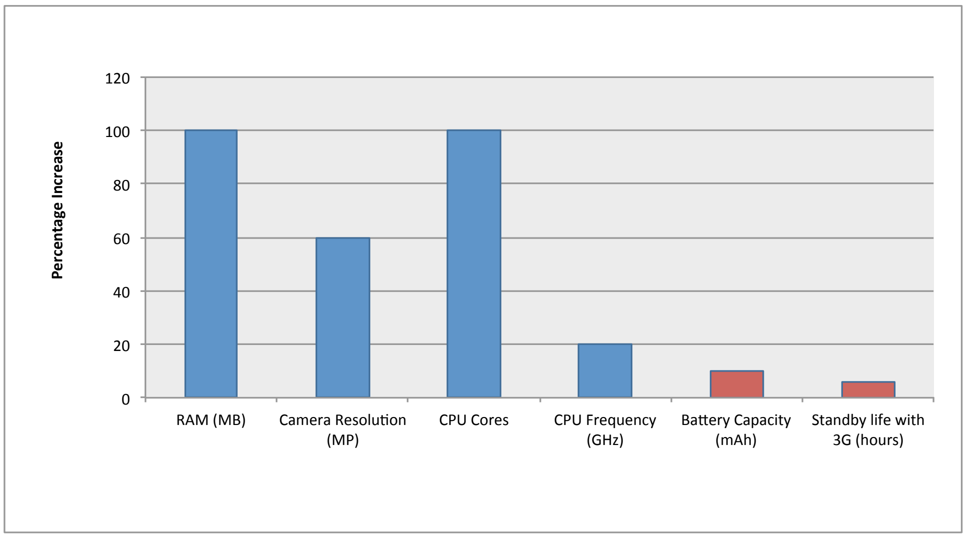


Figure 1.3: Percentage increase of resource capacities of the Samsung Galaxy SII mobile phone compared to the Samsung Galaxy SI.

1.4 Smartphones: Limitations and Challenges

Even though mobile phones are an attractive platform for conducting social psychological studies, they have limitations and pose many challenges, which we describe in this section.

1.4.1 Battery Limitations

Mobile phones have limited battery capacity and therefore energy should be expended judiciously by the applications. Moreover, the battery capacity of mobile phones has not been increasing at the same rate as the other components of phones like sensors, CPUs, memory etc. Figure 1.3 shows the percentage increase in resource capacities between two consecutive releases of a widely used smartphone, the Samsung Galaxy SI released in 2010 and the Samsung Galaxy SII released in 2011. We can observe that the memory and the number of CPU cores have doubled, and the camera resolution has increased by 60%, however, the battery capacity has increased by only 10% and stand-by life by only 6%. Moreover, Figure 1.4 shows that the energy density of the lithium-ion type of battery, which is the most commonly used in mobile phones, increased by only 2.3x from 1991 to 2005, whereas the increase in CPU speed during the same period is estimated to be more than 100x [INT08]. This means that the battery should be made larger in order to increase its capacity. However, since mobile phones are portable devices, battery size is an important consideration in their design.

The lithium-ion battery trends and the higher rates of energy consumption in mobile sensing applications (due to the powering of sensors) than in normal applications, motivate the design of power efficient techniques to perform phone sensing. Some research has been conducted in this sense [Nat12, KLJ$^+$08, WLA$^+$09].

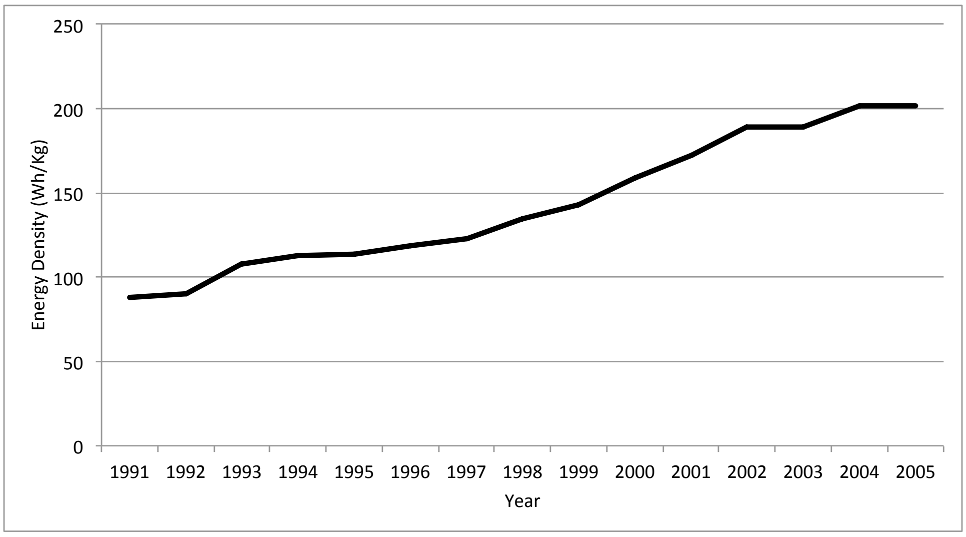


Figure 1.4: Increase in the energy density of Lithium-Ion battery from 1991 to 2005 (Source: http://batteryuniversity.com).

1.4.2 Processor Speeds

Although some modern mobile phones are equipped with many sensors, high-frequency and multicore processors, and large amount of memory, this is not the case with all smartphones. A large number of mobile phones have much lesser processing power and memory. For example, the HTC Wildfire, a top-selling basic smartphone, has fewer sensors, a single core 528 MHz processor and 384 MB memory, but the Samsung Galaxy SIII boasts a Quad-core 1.4 GHz processor and 1GB RAM. Accordingly, in order for mobile applications to scale to a large number of users using different phone models, social sensing applications based on mobile phones may need to exploit resources outside the phones, like cloud processing and using the sensors present in the infrastructure (for example, door, motion sensors) to provide the same level of service to all users. Exploiting remote processing and sensing resources also helps in reducing the phone's energy consumption by avoiding local phone processing and sensing.

1.4.3 Challenges

In addition to the battery and processing limitations there are also several other challenges posed by mobile phones, which need to be addressed to build social sensing systems.

- **Efficiency of sensor sampling.** Considering the battery limitations of mobile phones, the sensor sampling to capture data from the sensors of the phone cannot be performed continuously, as this will drain the battery rapidly. However, conservative sampling leads to the loss of valuable behavioural data and thus the behaviour of users may not be modelled accurately. Smart sampling schemes such as adaptive sampling that adapt the sampling to the user's context and achieve the required accuracy while conserving energy need to be designed.

- **Exploiting sensing infrastructure.** Although adaptive sensing schemes may help in reducing power consumption, they still need to spend energy in capturing the data (through local phone sensing). Modern buildings are instrumented with a variety of sensors such as RFID access control systems and light sensors. By offloading phone sensing tasks to infrastructure of sensors in smart-buildings, local phone sensing can be avoided thereby further increasing the energy savings. However, remote sensing imposes an increased cost in the form of network traffic. Thus, dynamic techniques need to be designed to exploit the sensors in the infrastructure of buildings considering the mobile patterns, sensing and network cost. In order to utilise the sensing infrastructure, mobile phones should be able to discover the services and sensor capabilities. Service discovery is a major challenge and there have been some works to this end [ZMN05, Dar10]. Various technologies have been used for achieving service discovery and advertisements such as Jini network technology, Microsoft's Universal Plug and Play (UPnP), and Service Location Protocol (SLP) [Ric00].

- **Accuracy of classifiers.** The sensors in mobile phones are not designed to capture the behaviour of users. The microphone sensor, for example, is designed for phone calls and not necessarily for speaker identification or mood recognition. Therefore, efficient and accurate classifiers have to be developed in order that accurate inferences may be inferred using raw data from the potentially inaccurate sensors of the mobile phone.

- **Computation offloading.** Certain classifiers such as for speaker identification [LBBP+11] or image/face recognition [CBC+10] are computationally intensive and will consume a large amount of energy if computed locally on the phone, as the processing takes a long time. Further, techniques like Natural Language Processing (NLP) can be applied effectively only with large dictionaries, which are better stored in the cloud. Local phone and cloud computing resources should thus be exploited to process the classification tasks effectively. However, use of cloud resources consumes the user's data plan and there are generally limits to the amount of mobile network data that can be used by the user in a calendar month. Moreover, wireless transmission of the data also consumes energy, so intelligent techniques should be developed to exploit the local phone and cloud resources while considering various dimensions like data plans, energy, and accuracy.

- **Privacy.** The data captured from the phone sensors is sensitive to privacy, for example, the voice data recorded for the speaker identification and the location captured from the GPS sensor. Therefore, privacy aspects of the sensor data should be considered in phone sensing systems. Microphone recordings can be deleted after extracting the essential features from the them. Bluetooth identifiers can be hashed using one-way Secure Hash Algorithms such as SHA-256. There have been some works on the privacy aspects of smartphones such as preserving the anonymity of sensor reports without reducing the precision of location data [TKFH06] and privacy-aware architecture for pervasive applications [CKK$^+$08].

- **Usefulness of applications.** In order to motivate users to participate in social studies, the applications should be useful. Therefore, using phone sensor data and inferences, social applications need to be built that not only provide useful data to the social scientists, but also provide value to users so that they continue to use the applications and participate in the experiments.

1.5 Thesis and its Substantiation

In the previous sections we discussed the advantages of using mobile sensing to capture the behaviour of users and conduct experimental social psychological studies, and we presented the limitations and challenges posed by this use of mobile phones.

Our thesis is that smartphone sensing can be used to automatically capture the behavioural and social aspects of the user, and can be an effective tool in the conduct of social studies.

Using smartphones in social psychology research involves i) designing and building software components on phones that can capture the user's behavioural data, ii) classify and draw inferences about behaviour, and iii) model the user's behaviour using these inferences and support social applications. Each of these components poses research questions that need answering to support smartphones based social sensing (Figure 1.5). In particular, the research questions posed by these components are as follows:

- **Research Question 1.** How can we accurately capture raw data from the sensors in smartphones in an energy-efficient way?
- **Research Question 2.** How can we efficiently process data captured through smartphone sensors to draw inferences about the user?
- **Research Question 3.** In what ways can smartphones be helpful in the conduct of social studies?

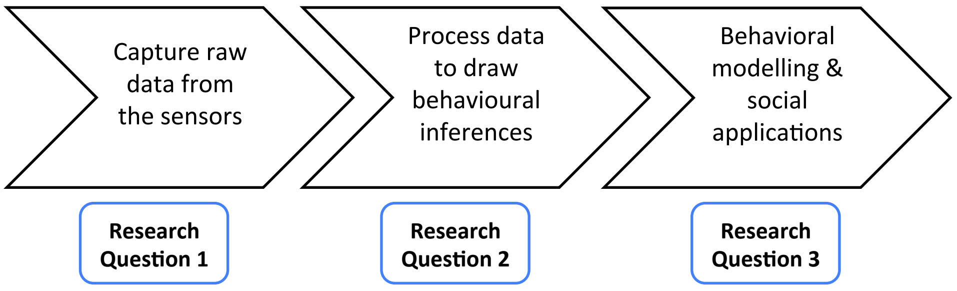

Capture raw data from the sensors
Process data to draw behavioural inferences
Behavioral modelling & social applications
Research Question 1
Research Question 2
Research Question 3

Figure 1.5: Components of social sensing systems.

In support of our thesis and to answer the above questions we explore techniques and models that enable efficient support of social sensing on smartphones (to answer Research Questions 1 & 2); Then, utilising these techniques, we design and deploy social psychological applications in real environments to demonstrate the usefulness of smartphones in the conduct of social studies (to answer Research Question 3). In particular, we explore the following: First, we design an adaptive sensor sampling framework that adapts the sensor data capture process to the user's context and achieves energy savings, while maintaining the level of accuracy required for the applications to be functional. Second, to save energy further, we design and evaluate a sensor task offloading scheme that efficiently uses both the local phone sensors and those in building infrastructure to achieve energy savings without compromising the accuracy of the system (We answer Research Question 1 by designing the adaptive sensing and sensing offloading schemes). Third, we design a computation offloading scheme that exploits local phone processing and cloud resources while considering the requirements in terms of energy, latency, and data traffic (We answer Research Question 2 through the design of the computation offloading scheme). Fourth, using these services, we design and deploy three social psychological applications to demonstrate the usefulness of smartphones in conducting social studies (We answer Research Question 3 using the findings from these deployments).

1.6 Contributions and Chapter Outlines

This dissertation explores the use of smartphones in performing social sensing and conducting social psychological studies. Figure 1.6 shows a high-level mapping between the chapters and the problems addressed. Our main contributions and chapter outlines are as follows:

- **[Contribution 1] Adaptive Sensor Sampling.**

  Sensor sampling is one of the fundamental components of any social sensing system that aims to capture data from mobile phone sensors. Sensors embedded in the mobile phone have to be sampled often in order to capture all the user's context

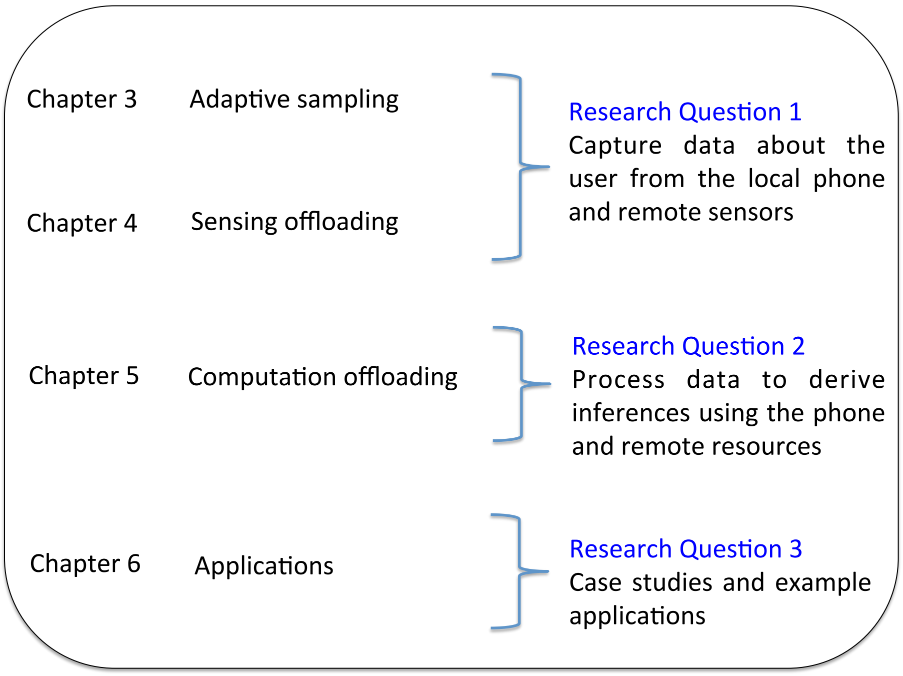

Chapter 3
Adaptive sampling
Chapter 4
Sensing offloading
Research Question 1
Capture data about the user from the local phone and remote sensors
Chapter 5
Computation offloading
Research Question 2
Process data to derive inferences using the phone and remote resources
Chapter 6
Applications
Research Question 3
Case studies and example applications

Figure 1.6: Chapter outlines and the proposed schemes.

events. This, however, leads to faster depletion of the phone's battery and may also discourage users from using these resource-intensive sensing applications as their first priority is to *make/receive phone calls* or *send/receive messages*. On the other hand, if the sensors are sampled at a slower rate, then the system may capture only a subset of the user's context events.

Sensor sampling schemes that achieve energy savings by adjusting (or lowering) the duty cycling rate of a sensor may not capture all context events. Therefore, we also need to measure the accuracy of sampling schemes to understand their performance. We use the following terms in our definition of accuracy:

– **Context events.** A context event is an event inferred from data sensed by a sensor, for example, in case of the phone's accelerometer sensor, it could be a "user stationary" or "user moving" event inferred using a movement detection classifier. We define total context events as the total number of events that can be inferred from a sensor data stream for a given length of time.

– **Interesting events.** A system might detect many types of context events using a sensor, however, not all of these events might be interesting to the given application. We define an interesting event as an event that is inferred by the given classifier and that is of interest to the application. E.g., considering the accelerometer sensor stream and a movement detection classifier, an application might only be interested in "user moving" events. We define total interesting events as the total number of events that are of interest to the application that can be inferred from a sensor stream for a given length of time.

We define the accuracy of a sensor sampling scheme as the number of interesting events detected by the scheme divided by the total number of interesting events occurred in the sensor stream for a given length of time. This is same as true-positive rate considering interesting events as positive cases. For example, considering a movement classifier based on the accelerometer sensor and assuming that an application is interested in capturing "user moving" events, if n "user moving" events have occurred in a given time period and if the system has detected n' of them, then the accuracy during this period is calculated as (n'/n). Since we only consider interesting events (true-positive rate) for the measurement of accuracy, a problem is that a scheme that reports events continuously would achieve high accuracy but may not be efficient to use on mobile phones. Even though we do not explicitly measure the number of uninteresting events detected, the proposed sampling schemes are all designed to minimise the number of event detections that are not of interest to the application. Further, another limitation is that the accuracy reported for a scheme also depends on the classification accuracy (for example, accuracy of a movement detection classifier in reporting "moving" and "stationary" events), i.e., considering the possible inaccuracies in classification, the accuracy of a scheme might decrease or increase due to errors in the classification process. Therefore, the performance reported by using this metric should only be viewed along with the type of context events and the classifiers considered.

We design an adaptive sampling technique that balances the energy-accuracy tradeoffs through the use of *linear reward-inaction learning* [BH75, KLM96] that is based on the theory of *learning automata* [NT89]. The adaptive sampling scheme adjusts the sampling rate of the sensors dynamically based on the user's context in terms of events observed (interesting or not), and thereby achieves energy savings without compromising the accuracy 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 evaluate the adaptive scheme by comparing it with a continuous sensing scheme and several function based sampling schemes. We provide a detailed description and evaluation of this scheme in Chapter 3.

- **[Contribution 2] Sensing Offloading.**

  Although adaptive sensing reduces the energy consumption of the sampling process, energy still needs to be spent for local sensing to capture the user's data. An approach that can further increase energy savings is to offload the sensing tasks to sensors in building infrastructure. Most modern urban buildings are already instrumented with sensors such as Radio Frequency Identification (RFID) access control systems. There is also an increasing trend to instrument buildings with a variety of sensors such as Passive Infrared (PIR) sensors, door sensors, and light sensors. However, we note that a considerable number of buildings may not yet have sensing infrastructure and in this case the phone can depend on its built-in sensors. Phone applications can exploit the presence of such infrastructure if available to reduce the energy impact of capturing social activities.

Allowing mobile applications to interact with infrastructure sensors whenever they are available provides an opportunity to design social sensing systems that can maintain accurate sensing using both the phone and infrastructure sensors without compromising the battery life. We therefore design a sensing offloading scheme that efficiently utilises the local phone and infrastructure sensors considering the phone sensing cost, the cost of communication between the phone and the sensing infrastructure, and the dynamic mobility patterns of the user. The presence of sensing technologies within most modern urban buildings offers frequent opportunities to apply this technique. For example, by relying on a building's RFID access control system, a phone application can suspend any localisation mechanisms on the phone while the user remains in the same room. In situations where appropriate sensors are not available in the infrastructure, the phone can fall back on traditional phone sensing techniques. Given the typical living patterns of most users, where a vast proportion of their daily lives is spent at their home or in a working environment, the sensing offloading approach may potentially achieve significant energy savings, thus further enhancing the acceptability of operating social sensing applications on mobile devices. In Chapter 4 we explore the offloading of sensing to infrastructure to achieve energy savings without compromising the accuracy of the applications. We also show that when sensing offloading is used along with the adaptive sensing scheme, energy savings are further increased.

- **[Contribution 3] Computation Offloading.**

  Once the data is sampled using the local phone and remote sensors it needs to be processed in order that high level inferences may be derived. 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 the microphone data or image recognition from the 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 have unlimited data plans. The allocation of the execution of computational tasks is thus vital in such systems.

  We design a computation distribution scheme based on *multi-criteria decision theory* [KR76] 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 energy, latency, and traffic trade-offs. We also design a rule-based framework to dynamically adapt the behaviour of the scheme with respect to changes in the mobile phone resources (like battery charge/discharge cycles, user's data plan running out of allowance). We note that when an intensive computation task is offloaded to the cloud, it may result in energy savings on the phone but it will consume energy on the cloud.

Throughout this dissertation, when we mention energy saving, the saving is on the mobile phone and not on the overall energy of both the mobile phone and cloud processing. We present the computation distribution scheme and its evaluation in Chapter 5.

- **[Contribution 4] Social Psychology Applications.**

  Once the adaptive sensor sampling, the sensing offloading, and the computation offloading components are in place, various social applications can be built using these services. We design and deploy three example social sensing systems: a passive behavioural sensing application, a collaboration and interaction detection application for the workplace, and an application that provides realtime feedback to users, to demonstrate the kind of data that can be collected and the analysis that can be performed using mobile phones and to show the advantages of the proposed schemes (Chapter 6).

  - **Example Application 1: EmotionSense**

    EmotionSense is a framework for collecting data in human interaction studies based on mobile phones. EmotionSense infers data on participants' *emotions* as well as proximity and patterns of conversation by processing the outputs from the sensors of off-the-shelf smartphones. Using this example application, we show that mobile phones can be used to understand the correlation and impact of interactions and activities on the emotions and behaviour of individuals. The key components include two subsystems for emotion detection and speaker recognition built on a mobile phone platform based on the Gaussian Mixture Model (GMM) [SRL03]. EmotionSense automatically identifies speakers and recognises emotions by means of classifiers running locally on off-the-shelf mobile phones. We evaluated the EmotionSense application through a real deployment involving 18 users over 10 days.

  - **Example Application 2: WorkSense**

    WorkSense is a social sensing application that utilises the sensing offloading scheme to achieve accurate sensing of social activities at the workplace. Using this example application, we show that mobile phones can be used to automatically detect the social interactions of users at the workplace by tracking formal and informal meetings during their daily routines and to infer how social interactions may affect their performance. WorkSense is able to detect various meetings and collaboration patterns of users at the workplace. We evaluated the WorkSense application with a real deployment within our research institution for about a month.

– **Example Application 3: SociableSense**

SociableSense is another example social sensing application that aims to provide realtime feedback to users to help them in fostering their interactions and improving their relations with colleagues. The application utilises the services of adaptive sampling and computation offloading schemes. The *social feedback component* in the application estimates the sociability of users (i.e., a quantitative measure of the quality of their relations) based on the interaction and colocation patterns extracted from the sensed data at run-time, and provides them with feedback on their sociability and strength of relations with colleagues. It also alerts users to opportunities to interact. In order to demonstrate the usefulness of SociableSense to the social scientists and participants, we conducted a social psychological study in an office environment where 10 participants carried mobile phones for two working weeks.

Contributions 1 and 2 are intended to answer Research Question 1, Contribution 3 is intended to answer Research Question 2, and Contribution 4 to answer Research Question 3. Many research questions in the field of social sciences can be further explored and validated using mobile phone sensing technology, given that the phones are: ubiquitous, unobtrusive, and sensor-rich devices. Research in mobile social sensing could also be further explored, given the findings in this dissertation. We shall present them at the end of the dissertation, in Chapter 7.

1.7 Publications

During my Ph.D., I have worked on the following publications that include workshop and conference papers, a book chapter, a demo paper, and a paper under review.

1.7.1 Publications Related to this Dissertation.

Conference Papers

- [LRMR13] Neal Lathia, Kiran K. Rachuri, Cecilia Mascolo, and Peter J. Rentfrow, Contextual Dissonance: Design Bias in Sensor-Based Experience Sampling Methods, in Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing (ACM UbiComp'13), Zurich, Switzerland, 2013.

- [REL$^+$13] Kiran K. Rachuri, Christos Efstratiou, Ilias Leontiadis, Cecilia Mascolo, and Peter Jason Rentfrow, METIS: Exploring Mobile Phone Sensing Offloading for Efficiently Supporting Social Sensing Applications, in Proceedings of the 11th IEEE Pervasive Computing and Communication Conference (IEEE PerCom'13), San Diego, California, USA, Mar, 2013. **[Won the Mark Weiser Best Paper Award]**

- [RMMR11] Kiran K. Rachuri, Cecilia Mascolo, Mirco Musolesi, and Peter Jason Rentfrow, SociableSense: Exploring the Trade-offs of Adaptive Sampling and Computation Offloading for Social Sensing, in Proceedings of the 17th Annual International Conference on Mobile Computing and Networking (ACM MobiCom'11), Las Vegas, USA, 2011.

- [RMM$^+$10b] Kiran K. Rachuri, Mirco Musolesi, Cecilia Mascolo, Peter Jason Rentfrow, C. Longworth, and A. Aucinas, EmotionSense: A Mobile Phones based Adaptive Platform for Experimental Social Psychology Research, in Proceedings of the 12th ACM International Conference on Ubiquitous Computing (ACM UbiComp'10), Copenhagen, Denmark, 2010.

Workshop Papers

- [RM11] Kiran K. Rachuri and Cecilia Mascolo, Smart Phone based Systems for Social Psychological Research: Challenges and Design Guidelines, in Proceedings of the 3rd International Workshop on Wireless of the Students, by the Students, and for the Students (ACM S3'11, co-located with ACM MobiCom 2011), Las Vegas, USA, 2011.

- [RMM10a] Kiran K. Rachuri, Mirco Musolesi, and Cecilia Mascolo, Energy-Accuracy Trade-offs in Querying Sensor Data for Continuous Sensing Mobile Systems, in Proceedings of the Mobile Context Awareness: Capabilities, Challenges and Applications Workshop (co-located with ACM UbiComp 2010). Copenhagen, Denmark, 2010.

Book Chapters/Magazines

- [LPR$^+$13] Neal Lathia, Veljko Pejovic, Kiran K. Rachuri, Cecilia Mascolo, Mirco Musolesi, and Peter J. Rentfrow, Smartphones for Large-scale Behaviour Change Interventions, accepted for publication in IEEE Pervasive Computing, Special Issue - Understanding and Changing Behavior, 2013.

- [RMM12] Kiran K. Rachuri, Cecilia Mascolo, and Mirco Musolesi, Energy-Accuracy Trade-offs of Sensor Sampling in Smart Phone based Sensing Systems, Mobile Context Awareness, Book Chapter, Springer, 2012.

Demos

- [Rac12] Kiran K. Rachuri, EmotionSense: Emotion Recognition and Social Sensing based on Smart Phones (Demo), in Proceedings of the 1st ACM Workshop on Mobile Systems for Computational Social Science (co-located with ACM Mobisys 2012), Lake District, United Kingdom, 2012.

Under Submission

- Kiran K. Rachuri, Christos Efstratiou, Ilias Leontiadis, Cecilia Mascolo, and Peter Jason Rentfrow, Smartphone Sensing Offloading for Efficiently Supporting Social Sensing Applications, submitted to Elsevier Pervasive and Mobile Computing, 2013 (extended version of our PerCom paper).

1.7.2 Other Publications

I have also worked on other papers during my Ph.D., some of which are works begun during my M.S. at the IIT Madras, and others through collaborations at the University of Cambridge.

- Christos Efstratiou, Ilias Leontiadis, Marco Picone, Kiran K. Rachuri, Cecilia Mascolo, and Jon Crowcroft, Sense and Sensibility in a Pervasive World, in Proceedings of the 10th International Conference on Pervasive Computing (Pervasive'12), Newcastle, UK, 2012.
- Kiran K. Rachuri, and C. Siva Ram Murthy, Energy Efficient and Low Latency Biased Walk Techniques for Search in Wireless Sensor Networks, Elsevier Journal of Parallel and Distributed Comp. (JPDC), 2011.
- Saamaja Vupputuri, Kiran K. Rachuri, and C. Siva Ram Murthy, Using Mobile Data-Collectors to Improve Network Lifetime of Wireless Sensor Networks with Reliability Constraints, Elsevier Journal of Parallel and Distributed Computing (JPDC), 2010.
- Kiran K. Rachuri, and C. Siva Ram Murthy, On the Scalability of Expanding Ring Search for Dense Wireless Sensor Networks, Elsevier Journal of Parallel and Distributed Computing (JPDC), 2010.

2 Mobile Sensing : Literature Review

As discussed in the previous chapter, smartphones have revolutionised the mobile application space: with high-end processors, large memory capacity, and many sensors, we can now build many types of applications some of which seemed science fiction just half a decade ago. In this chapter we present an overview of mobile sensing systems including smartphone sensing and their application domains. We also discuss the type of sensors embedded in modern smartphones and present the main features of a typical smartphone sensing system while providing details of the existing work. We limit the discussion to applications using the sensors of mobile phones, as this is the main focus of the thesis.

Chapter outline. Section 2.1 provides a brief history of mobile sensing and is followed by a description of the main types of mobile sensing systems in Section 2.2. In Section 2.3, we present the various sensors embedded in modern mobile phones and their typical usage scenarios, followed by the types of mobile phone sensing paradigms in Section 2.4. We then present the applications of smartphone sensing in Section 2.5, and general components in phone sensing systems in Section 2.6. Finally, we present a summary of the chapter in Section 2.8.

2.1 A Brief History of Mobile Sensing

Advances in sensing technology have led to the creation of many kinds of sensors that can measure various physical properties such as temperature, pressure, and light. In general, different types of sensors are required to measure different physical properties. Thermistors and thermocouples, for example, are used to measure temperature, and capacitive and resistive sensors or hygrometers are used to measure humidity [WD10]. Advances in technology have also led to the miniaturisation of sensors thereby creating an opportunity for mass scale deployment of sensors that could be mobile. The University of California Berkeley's SmartDust [KKP99] project created miniature sensor nodes termed *SmartDust* that could be deployed on a large scale for capturing data. The nodes are equipped with temperature, humidity, pressure, light intensity, tilt and vibration, and magnetic field sensors, and are integrated into a cubic inch package. The nodes are also capable of communicating over wireless, and forming self-organising networks. Many research centres have been born out of the SmartDust project[1], for example, the University of California, Berkeley WEBS (Wireless Embedded Systems), and UCLA's (University of California, Los Angeles) Center for Embedded Networked Sensing (CENS).

Miniaturisation of sensors has also resulted in user-centric sensing such as implanting sensors in the human body to monitor various physiological activities and building custom devices with embedded sensors that can be carried by users. For example, the MIT Media Laboratory's sociometric badge [LWA+08, KCHP08] (also known as a sociometer) is a wearable device with many embedded sensors and is automatically able to detect the extent of face-to-face interaction, conversation, co-location, and physical activity from the sensed data. Some other examples of purpose-built devices are the active badge system [WHFaG92] and the Electronically Activated Recorder (EAR). The active badge system can be used for indoor localisation using a wearable device that transmits an infra-red signal. EAR is capable of recording audio samples using an electronic device without any intervention from the user. Sensors were also then introduced in PDAs like the HP iPAQ, music players like the Apple iPod, and mobile phones like the Nokia N95 and the Apple iPhone. With the rapid increase in the adoption of phones and the number of sensors in them, it soon became possible to monitor many of the parameters that were sensed by custom devices or sensor nodes with mobile phones.

2.2 Types of Sensing Systems

In this section we describe in detail the various types of sensing systems mentioned in the previous section, and their application scenarios.

2.2.1 Wireless Sensor Networks

Wireless Sensor Networks (WSNs) [ASSC02, ZSLM04, CEP+07] consist of small battery operated mobile nodes that are equipped with computing, communication, and sensing

[1] http://robotics.eecs.berkeley.edu/ pister/SmartDust/

capabilities. WSNs are capable of self-organising and communicate via multiple hops. They typically consist of two types of nodes: sensor and sink. A sensor node is a device that captures data through the sensors and transfers this data to the sink node (typically a powerful device such as a computer) via multi-hop communication over many other sensor nodes. A sink node stores all the collected data, and in many systems, provides this to the outside world through a web-service API or similar mechanism. Sensor nodes are equipped with many sensors like temperature, humidity, microphone, and camera. Some of the commercially available nodes include MicaZ and TelosB [MOT13]. Sensor networks have applications in many domains such as volcano monitoring [WALW+06], environmental monitoring [MHO04], habitat monitoring [DEM+10], and civil and structural monitoring [XRC+04]. Wireless sensor network technology continues to evolve and is an active research area.

It is generally difficult to use sensor nodes in conducting human monitoring studies as they are less powerful in processing and memory capabilities than smartphones, and moreover, carrying sensor nodes is a burden and can be felt obtrusive by users. If sensor nodes are made more powerful by enhancing their processing power, then it results in increased cost for experiment designers. More importantly, there are no application stores or continuous connectivity to the Internet for deploying applications or communicating with the cloud. However, sensor networks are now increasingly deployed as part of sensing infrastructure in buildings for various purposes (e.g., door, motion sensors), so they seem to provide a perfect platform for the augmentation of mobile phone sensing to save energy and improve accuracy. In Chapter 4 we discuss in detail a possible framework for mobile phone and sensor network sharing.

2.2.2 Body Sensor Networks

A Body Sensor Network (BSN) [Yan06, ZLL+11] is a wireless network of implantable and wearable sensors attached to a human body. They are typically used for monitoring various activities inside the human body, differ from traditional Wireless Sensor Networks in communication ranges, sensing tasks, and deployment scale, and are mainly used in healthcare. Moreover, a node in a BSN is typically biodegradable and biocompatible, whereas a WSN node may not be. A sensor node in a BSN captures a variety of physiological data from the human body such as heart rate, blood pressure, body temperature, and glucose levels. BSNs are used for a purpose different than those of the target application scenarios of the current dissertation, i.e., social and behavioural psychology, and social sensing. BSNs find applications in many scenarios in the domain of healthcare and wellbeing, for example, monitoring diabetic patients, automatic monitoring of patients in hospitals and the elderly, and post-operative monitoring. BSNs also find applications in the analysis of sports performance. Possible integration of the work presented in this dissertation and the Body Sensor Networks could be envisaged.

2.2.3 Wearable Sensing

In Wearable Sensing [PEK$^+$06, Bon03, CMP00, FMT$^+$99], wearable components such as wrist watches, jackets [FMT$^+$99], and badges [LM02, CBC$^+$08, ALC06] are embedded with sensors such as 3-axis accelerometers, cameras, and microphones to detect various activities of the user. In [LM02] the authors use biaxial accelerometer sensors, digital compass sensors, and angular velocity sensors to detect human activity, context, and location using movements in 2D environments. The Microsoft SenseCam [ALC06] is a wearable device equipped with camera, temperature, light, infrared, and accelerometer sensors. One of the main applications of SenseCam is the capture of a blog or diary of the user using various pictures taken by the camera. The other sensors are used for the efficient triggering of the camera (e.g., triggering based on the user's interaction detected through the IR sensor).

Although wearable sensors are extremely useful for automatic detection of context, and as shown, for example, in [LM02], accuracy is also high, these systems are only suitable for laboratory studies or limited deployments and are not scalable due to high costs of building hardware. Nor is it practical to ask users to use the wearable components in their day-to-day activities for prolonged periods.

2.2.4 Phone Sensing

Mobile phone based sensing can perform tasks similar to wearable sensing as sensors like accelerometer, camera, and microphone are already present in off-the-shelf phones. More importantly, they are already carried by many people, which makes their deployment highly scalable. Using phones also reduces deployment costs. We have discussed some of the details of the sensors and advantages and limitations of phone sensing in the previous chapter, and in the rest of this chapter we focus on the sensors in phones, the application domains of phone sensing, and general features of smartphone sensing systems.

2.3 Sensors in a Smartphone

Modern mobile phones such as the Apple iPhone and the Samsung Galaxy SII (Figure 2.1) have many sensors embedded in them. In this section we provide details of these sensors and their common uses.

- **3-axis Accelerometer.** Provides the X, Y, Z axis values of the triaxial accelerometer sensor. The accelerometer sensor in a phone is used mainly in games to enhance the user's experience. The accelerometer values can also be used to calculate the acceleration of the user, however, classifiers need to be developed to accurately infer activities like moving, running, and walking from raw accelerometer data [MLF$^+$08].

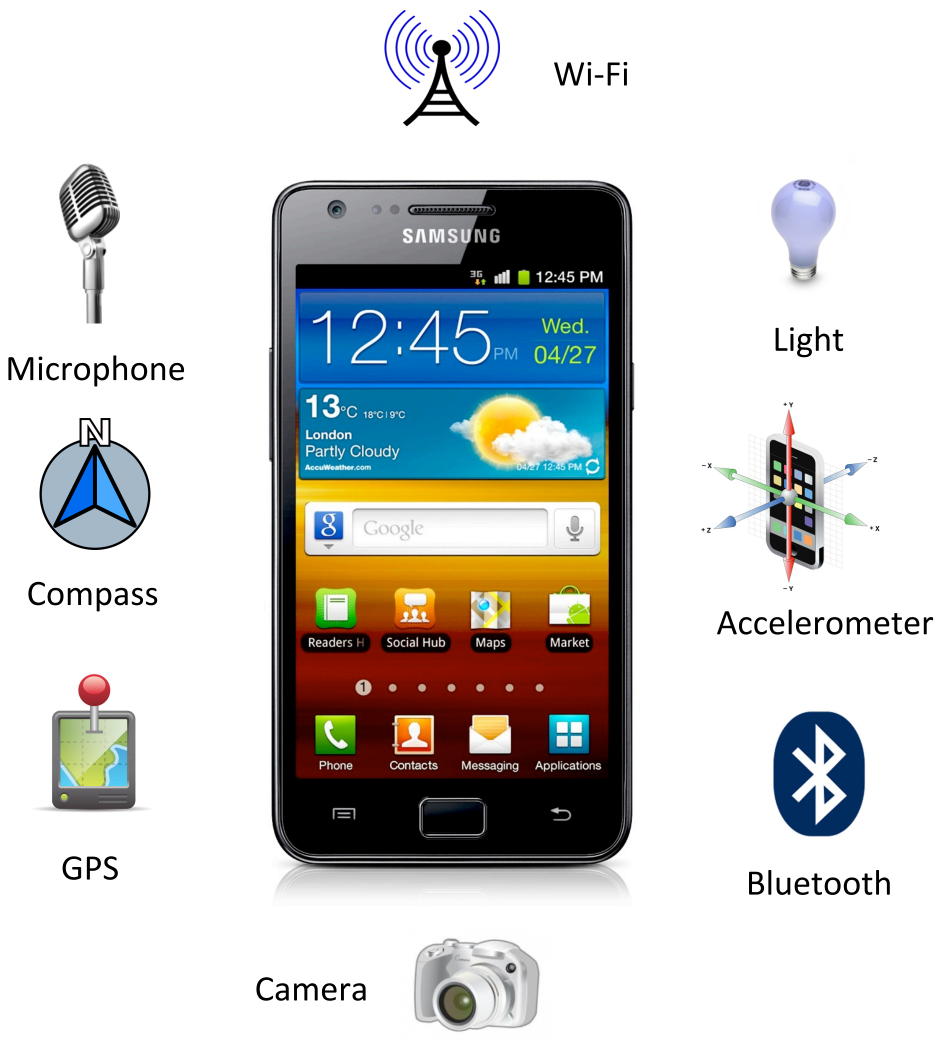


Figure 2.1: Sensors embedded in the Samsung Galaxy SII mobile phone.

- **Barometer.** Some modern mobile phones like the Samsung Galaxy S III [SG3] have a barometer sensor embedded. This sensor can be used to track changes in atmospheric pressure and to estimate weather conditions and altitude.

- **Bluetooth**[2]**.** The Bluetooth sensor can be used to discover nearby Bluetooth devices that are in discoverable mode[3]. It can be used, for example, to detect people co-located with the mobile phone user. Bluetooth sensing can also be used to perform indoor localisation by placing Bluetooth anchors at various locations and maintaining a static map of the anchors and locations. Bluetooth sensor has been

[2]Bluetooth is a radio and not a sensor like accelerometer or microphone. However, since data can be captured from this radio similarly to the sensors, in this dissertation we refer to the radios such as Bluetooth, GPS, Wi-Fi as sensors.

[3]Discoverable mode allows a Bluetooth device to be discovered by other nearby Bluetooth devices when they perform a scan. If a device is not in discoverable mode, then it will be hidden from the scan requests of nearby Bluetooth devices.

used by many works [BAB+10]. In modern mobile phones, the Bluetooth discoverability automatically switches off and may require periodic manual consent by the user to continue in the discoverability mode. This poses problems for co-location or indoor location detection. In these cases, the applications may have to depend on "Bluetooth anchors" in the environment to localise the users to infer their co-location.

- **Camera.** The camera can be used for taking pictures, video calls, and scanning QR codes (Quick Response Codes)[4]. In addition, it is shown in [WCC+12] that the phone's front and rear cameras can also be useful to alert the user to unsafe situations while walking on a road.

- **Compass.** The compass sensor can be used to detect the orientation and direction of the phone. It is used in navigation applications such as Google Maps [GMP].

- **GPS.** The Global Positioning System (GPS) sensor can be used to localise the mobile phone using satellite information. GPS generally has about metre-level accuracy but works only outdoors when there is a clear view of the sky. The localisation feature is used in many applications, for example, location-based social networks like Foursquare[5] use location information to show nearby places of interest to the user.

- **Gyroscope.** The gyro sensor is useful to measure the phone's orientation and is used for gesture recognition in gaming-related applications [ZCCM12]. Again, classifiers that can infer various user positions from raw sensor data need to be developed to effectively use this sensor.

- **Microphone.** This is a fundamental sensor and is present in every mobile phone. The microphone sensor can be used to identify the speaker, detect the user's emotions, and to determine noise levels in the environment. Noise in the user's environment affects the accuracy of techniques like speaker identification. Further, microphone generates a large amount of data due to its high hardware sampling rates, for example, recording a 5 seconds audio sample on the Android platform in PCM format generates a file size of about 100KB, which poses data management and processing challenges. Some applications have also used it for localisation [ACRC09].

- **Near Field Communication (NFC).** The NFC chip in a phone can be useful for short-range communication between similar devices usually no more than four centimetres apart. It finds application in peer-to-peer communication and mobile contact-less payments.

[4]https://itunes.apple.com/gb/app/qr-reader-for-iphone/id368494609
[5]https://foursquare.com/

- **Photo.** The photo or light sensor is useful to detect the ambient light levels in the environment of the phone. This sensor can be used to enhance the user experience by automatically adjusting the screen brightness.
- **Proximity.** This sensor is useful to detect whether an object is blocking the surface of the mobile phone. It is generally used to switch off the screen of touchscreen phones when the user is speaking on the phone.
- **Screen.** The screen of the mobile phone can also be used as a sensor to detect whether the user is interacting with the phone. It could be used in experience sampling techniques as a "trigger" to ask the user to complete a survey [FCC+07].
- **Wi-Fi/Cellular.** We may regard the Wi-Fi and cellular radios in mobile phones as sensors. These radios can be used to perform localisation [ALM]. Wi-Fi can also be used to detect co-location of users based on fingerprinting [BP00] or signal strength analysis.

These sensors in smartphones provide an excellent platform on which developers and researchers could build social sensing applications. In addition to their usage scenarios, the sensors also vary in power consumption: some are expensive power-wise, e.g., GPS in active state [ZTQ+10], and some are cheaper, e.g., accelerometer [SP12].

In addition to these sensors, data streams in smartphones such as Facebook status messages, Twitter data, Foursquare check-ins, Google calendar, communication patterns and application usage [LLLZ13], can also be considered as sensors that could also be a source of information about the user's context. However, the information provided by them may not always be reliable. For example, in [LOIP10], it has been reported that the calendar may not accurately represent the user's real context as the actual events are hidden by many placeholders and reminders. The unreliability of this information has also been discussed in [VF11], where the authors discuss the problem of "checking-in" to locations on Location Based Social Networks (LBSN) such as Foursquare without being physically present or forgetting to "check-in" in some places. They present a solution based on the phone's sensors and radios to validate the check-ins and to alert users to check-in. From these works, it could be inferred that the data from these "virtual" or "software" sensors may not always be reliable for context-inference and it may be necessary to depend on physical sensors.

2.4 Categories of Smartphone Sensing

Mobile phone sensing can be broadly divided into two [LML+10] categories: participatory and opportunistic sensing.

Participatory Sensing

In the participatory sensing model [BEH$^+$06] the mobile phone user is actively involved in the data collection process and the context is explicitly given to the system by the user, for example, taking a picture or recording audio data from the microphone sensor when at a particular location. The data is deliberately collected by the user upon a pre-defined criterion, like collecting an air pollution sample when at a particular location and time. The main advantage of this model is that the sensing is driven by the user, and the applications need not perform continuous context sensing, thereby saving energy. In a way, the complex context recognition step is avoided by leveraging the intelligence of the user [BEH$^+$06]. The disadvantages are that the data capturing part is dependent on the user more than on the machine (phone) and therefore some data samples might be missed if the user forgets to act, compromising the accuracy or functionality of the applications. Moreover, since this model uses the user's time, it will be a burden on the user to collect data manually. Accordingly, incentive mechanisms [LH10] should be designed to retain user participation levels. Participatory sensing systems include that presented in [ZZL12], which predicts the arrival times of buses from data collected by bus passengers, and the LiveCompare system [DC09] which facilitates inter-store grocery price comparisons using photographs of price tags taken by the participants.

Opportunistic Sensing

In the opportunistic sensing [LML$^+$10] model, data from the sensors in the mobile phone is captured passively and the context is inferred automatically by the device and the user is not involved in the data capturing process. An example is collection of accelerometer samples [MLF$^+$08] automatically to classify the user's activity, or audio samples [MCR$^+$10] to identify the conversation status. In this model, since the user is not involved, the burden on the user will be less and the user may not feel as obtrusive as in the case of the participatory sensing model. Further, since the data capturing process is automatic and is performed by the phone, it will not miss any sensor samples if sensing is continuous. However, continuous context recognition requires continuous sampling of sensors, which results in rapid depletion of the phone battery [WLA$^+$09].

In this dissertation, we focus on the opportunistic sensing model where the data capturing process and the context recognition is automatically performed by the phone without the user's involvement.

2.5 Applications of Smartphone Sensing

Smartphones find applications in many domains such as activity recognition, transport and navigation, entertainment, and social psychology. In this section we discuss some of the application domains and in particular focus on the works that use sensors embedded in them.

2.5.1 Activity Recognition

Smartphones can be used to track the physical activities of users like running, cycling, and walking, using the accelerometer sensor [MLF+08, PPC+12, ZCCT12]. This is achieved by capturing [x, y, z] axis vectors of the accelerometer sensor and classifying them using pre-trained models for physical activities. The movement speed can be calculated using readings from the GPS sensor. These inferences can further be used to provide useful services, such as fall detection for the elderly [YKE+10], or to enhance the social network experience [MLF+08]. The CenceMe [MLF+08] system captures activity data on walking, running etc. and the conversation status of the user using a Nokia N95 smartphone, and automatically updates the user's social network status. For activity recognition, they achieve an accuracy of about 94% for walking and about 74% for running. The EEMSS [WLA+09] system is a mobile phone system implemented on the same smartphone model (N95) and is able to detect various activities like working, meeting, and being in a vehicle. In [PPC+12], the authors present a technique to perform device's pose detection and the user's walking speed estimation. They show that the scheme achieves an accuracy of 94% for pose classification. Inferring the pose of the phone, for example, can be used in pollution sensing applications as inferring that the phone is inside a bag is useful to turn-off resource-intensive pollution sensor and for accurate labelling of data. The work presented in [ZCCT12] addresses the problem of reliably estimating cycling activity using a smartphone. By combining the GPS traces with data from the US Geological Survey elevation service, and OpenStreetMap database, they achieve an accurate estimate of caloric expenditure (they achieve a Root Mean Square error of about 5). Other applications like Endomondo[6] and runtastic[7], available in the Apple App Store and the Google Play, can be used to track running using the GPS and accelerometer sensors.

2.5.2 Transport

Vehicles can be instrumented with sensors and given that they have more capabilities and resources than phones, they can be used to monitor traffic conditions and travel estimates. MIT's pothole patrol system uses vehicles equipped with accelerometer and GPS sensors to detect potholes in roads. However, achieving similar functionality using off-the-shelf mobile phones has the benefits of reducing the cost as not all vehicles are equipped with the required sensors and improving the scalability as mobile phones are used by millions

[6]http://www.endomondo.com

[7]http://www.runtastic.com

of users in any city, but the same can not be said of vehicles, especially in some cities. In NeriCell [MPR08] the authors designed a mobile phone based system to monitor potholes, bumps, and honking using the accelerometer, microphone, and GPS sensors. Smartphones can also be used to estimate travel time, as demonstrated in [TRL$^+$09] using the Wi-Fi, GPS, and GSM (used for cellular triangulation) sensors. They show that the travel estimates just based on Wi-Fi localisation are good enough, by demonstrating that 90% of commute paths found were no worse than 10-15% of the optimal path. They also show that by combining Wi-Fi with GPS improves the estimates only by a small margin, but with a large increase in the energy consumption.

2.5.3 Entertainment and Games

One of the most popular application categories for mobile phones is games. For instance, 8 out of the top 10 most downloaded iPhone apps belong to the games category, which shows their wide use [GIZ12]. Sensors in the phone can be used to enhance the gaming experience of the user. In [ZCCM12] the authors used the microphone and accelerometer sensors to support phone-to-phone mobile motion games. They performed a case study using a game called SwordFight, a sword fight duel between two users where each user's phone simulates a sword. The authors of [MML$^+$08] showed that the divide between the virtual world and the real environment can be bridged and user experience can be enhanced using smartphone sensing.

2.5.4 Enhancing User Experience

User experience in interacting with electronic devices and performing digital tasks can be enhanced by using the sensors in smartphones. This enhancement could be as simple as rotating the screen by detecting changes in the phone's position using the accelerometer sensor, or disabling the caller tone based on pre-configured gestures. In Switchboard [MAZ$^+$11] the authors presented a matchmaking scheme in mobile multiplayer games over cellular networks. They achieve a match that reduces the burden on cellular networks and phones while satisfying the latency requirements of games, for an enhanced user experience. Point & Connect [PSZL09] is a device pairing scheme that uses the microphone sensors of the phones to pair the devices, which would otherwise take multiple steps: discovering nearby devices, selecting the target device, authenticating, and then pairing. As the name suggests, when the user needs to pair his/her mobile phone with another in proximity, then he/she simply points the phone at the target phone. The system captures the user's gesture and completes the device pairing by using acoustic based distance-change detection. Sensors can also be used to speed up the application launching process as demonstrated in [YCG$^+$12]. The authors use the user's contextual information such as location, obtained using the GPS sensor, and temporal access patterns to predict application usage and prelaunch applications. They show that the scheme reduces application launch delay by around 6 seconds on average.

2.5.5 Social Psychology

Recently there has been a lot of interest in applying the concepts of mobile phone based sensing to social psychological sciences. In this subsection we discuss this application domain in detail as it is the main focus of the dissertation. At a high level, we can divide social psychological mobile applications into two types: passive monitoring to study user behaviour, and feedback techniques to study the effect of interventions and incentives on behaviour.

Behavioural Monitoring

Traditional methods of social psychological studies for behavioural monitoring are based on collecting data from users through self-reports, which involve the users' reporting details about their mood, conversations, location, and activity periodically, for example, daily or weekly. However, it is hard for the users to accurately remember [Tou99] their experiences, and it is much harder to remember emotional states in detail [MP01, Bla86]. To reduce memory errors, a better method called *Experience Sampling* [CL87] was designed, which asks participants to complete self-reports several times a day by providing an alert/notification via electronic devices such as pagers. Ecological Momentary Assessment (EMA) [SS94] also involves signalling participants to report on their current or recent psychological and behavioural states multiple times a day using electronic devices like palmtop computers. Assessments can be on immediate affective, cognitive, or behavioural experiences, or on longer time events. However, self-report methods and behavioural sampling methods can be felt as a burden on the user as he/she has to take time to accurately recollect behavioural states. Furthermore, self-reports and experience sampling methods have been found to be biased towards positive or pleasant experiences [PR91].

To overcome the problems in self-reporting and experience sampling methods, Mehl *et. al.* proposed the Electronically Activated Recorder (EAR) [MP01], a device that captures audio samples in the user's environment using an audio recording device for 30 seconds once every 12 minutes. The audio files are transcribed offline by social scientists who manually listen to them. This method has some advantages over self-reports as it avoids memory errors and captures the real behaviour of the users without bias (as participants are not required to complete behavioural reports). However, there are also disadvantages. First, it takes a considerable manual effort from social scientists to transcribe audio files. Second, the purpose-built audio recording device must always be carried by participants and can be very obtrusive. Third, since audio files are accessed by people other than the user, this could raise privacy concerns.

Smartphones have the potential to overcome these problems as they are equipped with sensors to capture data about the behaviour of participants. Further, they are equipped with powerful CPUs, therefore, classifiers can be run on them to draw inferences about user behaviour thereby reducing the manual effort of coding and also reducing the burden of completing self-reports on participants. Moreover, phone systems can be designed to delete the raw sensor data immediately after classification thereby preserving the privacy of participants. Recently researchers proposed mobile phone based systems [LRC$^+$12, LLLZ11] to capture user behaviour. StressSense [LRC$^+$12] is a mobile sensing system based on smartphones that uses audio recorded through the microphone sensor to detect stress in the user's voice. The authors use many acoustic features from the recorded audio for classification such as standard deviation of pitch, perturbation, and speaking rate. They show that the system achieves an accuracy 81% and 76% for indoor and outdoor environments, respectively. In [FCC11], the authors present a voice classification library on smartphones and show that it is feasible to perform real-time voice classification on off-the-shelf phones, and the authors of [LBBP$^+$11] present an energy efficient speaker identification scheme using the microphone sensor. They show that an accuracy of over 90% can be achieved using a 3 second sampling window and 8kHz hardware sampling rate. Voice classification and speaker identification have many applications in the field of social sciences, such as detecting speech and interaction patterns of the users. The MoodSense system [LLLZ11] infers the user's mood from the information such as SMS, email, and browsing history already available in modern smartphones. They show that a generic clustering scheme (one-size-fits-all model) can achieve an accuracy of 61% (considering four emotion categories), and a subject-specific scheme improves the accuracy to 91%.

Feedback

Feedback on activities and behaviour helps users in understanding their own patterns. Mobile phones are an excellent platform to provide various kinds of feedback to users as they are ubiquitous and sensor-rich. More importantly, they are frequently accessed by the users [DBB11]. Feedback can be designed to provide useful statistics to the user or for individual well-being like motivating the user to drink healthy quantities of water [CCC$^+$09], or community well-being like encouraging water conservation [ABS05], and green transportation [FDK$^+$09].

Many research projects have used mobile phones to provide feedback: TripleBeat [DOO08] is a platform that assists runners to achieve their exercise goals and uses a virtual competition with other runners to encourage users. Some systems have also included social gaming features like leader boards or rankings, or updating the user's social network. For example, Foursquare[8] is a widely-used mobile phone application that provides a location-based social network service. Users can *check-in* to the locations they visit and receive points that can be compared with those of the friends.

2.6 Components of Smartphone Sensing Systems

In this section we present the general components of smartphone sensing systems and discuss the existing work.

2.6.1 Sensor Sampling

Sensor sampling is one of the fundamental components of mobile sensing systems [LML+10]. Raw data is captured from the sensors in a phone using the sensor sampling component. The format of raw data differs for each of the sensors. For example, when a 3-axis accelerometer sensor is queried the raw data is in the form of [x, y, z] vectors, and the raw data from the microphone is an audio file of the form *.wav*, *.3gpp*, or *.pcm*.

Sensors in current off-the-shelf smartphones can be divided into two types:

- **Pull Sensors.** In this type, the sensor should be queried periodically to capture data. For example, in the Android system, the Bluetooth sensor should be periodically scanned to detect the user's co-location, and audio data should be recorded from the microphone sensor often to detect conversation status. Sensors like microphone, Bluetooth, accelerometer, GPS, and Wi-Fi fall within this category.
- **Push Sensors.** In this type, the sensor needs hardware-level support to efficiently perform push notifications, for example, screen on/off notifications. The modules interested in a sensor data stream should register or subscribe to updates from the sensor, and they will be notified of new events by the operating system[9]. For example, in the Android system, developers should implement *SensorEventListener* and register with *SensorManager* to receive updates from the proximity sensor. Sensors like proximity and screen (on/off) are some examples of push sensors.

Data can be gathered from the sensors in the mobile phone either by querying periodically or through notifications. The push sensor data collection process is already efficient as

[8]https://foursquare.com

[9]A push sensor that is efficient requires hardware-level support and will typically be accessed through a publish/subscribe mechanism. However, if a sensor's data is accessed through a publish/subscribe method then it is not necessarily a push sensor. For example, in the Android system, data from the accelerometer sensor in the phone is accessed using a publish/subscribe mechanism, however, the operating system queries the sensor periodically to access its data and notifies subscribers. We do not consider this type of sensors as push sensors.

it can be captured by the corresponding hardware, which then notifies the operating system of this event (e.g., screen on event triggers a notification to the operating system). However, pull sensor data collection is challenging because the sensor has to be queried periodically to detect new events. The rate at which the sensor is queried will affect the accuracy, energy, and latency of the system. Therefore, there is an opportunity to design smart sensor sampling techniques to capture data from the pull sensors while balancing energy-accuracy trade-offs.

2.6.2 Data Processing

Once the raw data is captured by the sensor sampling process, it needs to be processed to permit inferences. For example, immobility and actions like driving, running, and walking can be inferred from the raw [x, y, z] vector data of the 3-axis accelerometer sensor. These inferences are generally based on machine learning techniques [LRC+12, Bis07]. The process involves two steps, *feature extraction* $\rightarrow$ *classification*. Some of these classification tasks such as detecting whether the user is moving or stationary are trivial and do not require much computing power, whereas others such as speaker identification [LBBP+11] based on microphone data, or face recognition [CBC+10] based on camera data are computationally intensive. Even though modern mobile phones have high computing power, using these computing resources for intensive classification tasks consumes a lot of energy [LBBP+11]. Furthermore, the computing requirements of some classification tasks may just be too high for the phones. Typical classification tasks in mobile systems include an activity recognition [MLF+08] to detect the type of physical activity of the user like running, walking, cycling, or standing, and face recognition [CBC+10] to tag users in an image (however, alternate techniques based on other sensors can also be used for image tagging as shown in [QBRCN11]). A speaker identification [LBBP+11] classification task is used to identify who is speaking, and finds applications in social psychology, and a stress detection [LRC+12] classification task can be used to detect the psychological wellbeing and stress experienced by users.

2.6.3 Applications/Information/Interventions

Using the sensor sampling and classification components, phone systems can infer the user's status, e.g., that the user is in conversation with person A at the *cafeteria* or the user is driving *home* from the *office*. An important goal of these systems is to use these inferences to build useful applications for end-users. This could be as simple as just providing some useful statistical information about the user's activities like length of time spent in conversations, or updating the user's social network status with his/her current activity [MLF+08]. More advanced application scenarios are to design and provide interventions or persuasion methods such as motivating the user to drink healthy quantities of water every day [CCC+09], keep in touch with friends [Gol04], or to assist runners in achieving exercise goals [DOO08].

2.7 Energy Efficiency

Since mobile phones draw power from a battery source, the energy efficiency of phone applications is an important and a pivotal design consideration. It is achieved by adopting various techniques, some of which are detailed in this section.

Duty Cycling

Duty cycling is the process of limiting the length of time a sensor is active. If the duty cycling of a sensor is 10% then it senses data only 10% of the time. Duty cycling is a widely-used technique in Wireless Sensor Networks [BYAH06, CAHS05, LKR04] and also in many mobile phone sensing systems [WLA+09, MLF+08, LYL+10, KLGT09, SP12].

The main reason for performing duty cycling is to save energy by powering off the sensor periodically. Even though, this might save energy, there is an additional cost incurred due to tail energy consumption [PHZ+11, PHZ12, BBV09], which is the cost incurred when a component remains in a powered state after it has been turned off. Tail energy costs for a sensor are typically less than the cost when the sensor is active [PHZ12]. It has been shown in [PHZ+11] that components such as wireless NICs, SD card, GPS have tail energy states. For example, they show that the HTC Tytn 2 phone's NIC continues to be in active state for about 1.7 seconds after a send/receive is completed. Some sensors also continue to be in powered state (tail state) after they are turned-off by the application, and they show that this duration is about 3 seconds for GPS. For example, if the application uses a GPS sense window of 30 seconds, the energy estimation based on only the active sensing periods could potentially lead to approximately 10% or more error.

In our evaluation of sampling schemes, we estimate the energy consumption based on the total active time of a sensor. As discussed, the limitation of this approach is that the tail energy states are not considered. Therefore, the performance reported using this method may only represent an approximate estimate of the absolute energy consumption. However, our main goal is to not measure the absolute energy consumption of the schemes, but to understand the relative benefits of the duty-cycled sampling schemes, therefore, this method of estimation allows us to estimate the relative performance differences among duty-cycling schemes. We, however, note that a limitation of our energy results reported for sensor sampling schemes is that they do not consider tail energy states.

Another disadvantage of duty-cycling schemes is that there is the possibility that events might be missed when the sensor is not active, resulting in loss of accuracy. In general, the lower the duty cycling, the higher the energy savings and the loss in accuracy.

Sensor Sharing

The sensor sharing approach relies on sharing the sensors of the phone with other phones in proximity, and opportunistic use of sensors in nearby phones. For example, if a mobile phone's battery level is low and if it needs location information then sensing from the GPS, which is an expensive sensor, is not energy-efficient. The phone could, therefore, request location information from the GPS sensor of a nearby mobile device, assuming communication with that device is cheaper. Likewise, when the phone's battery level is high it can then assist other mobile phones that require sensor data. It should be noted that, these type of techniques may reduce the energy consumption on the phone but incur more energy consumption elsewhere, for example, on the remote mobile device or the sensing infrastructure. This model of sharing by cooperation might save energy on the phone and has been used in some mobile systems [VRC11, LJM$^+$12]. In ErdOS [VRC11] energy savings are achieved by proactively managing resources and by exploiting opportunistic access to resources in nearby devices using social connections between users. CoMon [LJM$^+$12] 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.

Context Awareness

The user's context can be used to achieve high energy savings. For example, the GPS sensor can be turned off when the phone detects that the user is at *home* or at the *office* and it can be switched on when it detects that the user is moving. The Acquisitional Context Engine (ACE) [Nat12] is a middleware for mobile sensing applications that supports continuous context recognition while achieving high energy savings. ACE automatically learns the relations between various context attributes and then exploits these for further optimisations: *inference caching* and *speculative sensing.* Inference caching allows the platform to infer one context attribute from another already known attribute, e.g., the user is not at home when driving. Speculative sensing is used to find the value of an attribute by using cheaper proxy attributes. For example, the attribute: *whether the user is at the office* can be answered false, if the attribute: *whether the user is driving or running* evaluates to true. This can potentially save energy if the evaluation of proxy attributes is cheaper than the main attribute. Other example of systems in this category include SeeMon [KLJ$^+$08] and the Jigsaw continuous sensing engine [LYL$^+$10], which support continuous sensing while achieving energy efficiency.

Triggered Sensing

In triggered sensing low-powered sensors are used to trigger the sensing of a high-powered sensor. For example, the accelerometer sensor consumes much less energy than the GPS sensor. Therefore, given movement detection by continuously monitoring the accelerometer sensor, the GPS sensor can be activated/deactivated, thereby saving a significant amount of energy. The same was demonstrated in the SenseLess [BAPH09] system where it has been shown that this scheme can reduce energy consumption by more than 58%. In [PKG10] the authors present the Rate Adaptive Positioning System (RAPS), an energy-efficient localisation technique for smartphones. The system uses the history of the user to turn on/off the GPS sensor according to an estimated position accuracy and a given threshold value for accuracy. It uses the accelerometer sensor and Bluetooth communication to reduce position uncertainty, and exploits celltower-RSS blacklisting to detect GPS unavailability to avoid switching on the GPS.

Computation Offloading

Modern mobile phones have high-end and powerful processors, e.g., the Samsung Galaxy SIII is powered with a quad-core 1.4 GHz Cortex-A9 processor. However, using this processing power consumes a lot of energy [LBBP+11]. A viable option is to use cloud processing to save energy while trading off communication costs. Some mobile systems like [Kri10, KPKB10, CIM+11] exploit this feature to save energy. MAUI [CBC+10] supports fine-grained code offload by considering the current connectivity constraints of the mobile phone in order to achieve energy efficiency. The authors showed that by using code offloading MAUI saves 27% energy for a video game, 45% energy for a chess game, and much more for a face recognition application. They also showed that exploiting remote computing significantly reduces the latency of these tasks: the latency of performing a face recognition task is reduced from 19 seconds to less than 2 seconds. ThinkAir [KAH+12] is also a framework for offloading mobile computation to the cloud. The system accommodates changing computational requirements based on on-demand VM resource scaling, and exploits parallelism for faster execution.

Comparing with the existing work, the techniques that will be presented in this dissertation provide a novel way of achieving energy efficiency in capturing and processing data from the phone's sensors. The schemes achieve energy efficiency by using adaptive techniques that dynamically adjust the duty cycling interval of the sensors, and by exploiting the sensors in the environment and the computing resources in the cloud while considering the mobility patterns and the requirements of the user. Further, the schemes presented in this dissertation are orthogonal to most of the existing techniques that achieve energy efficiency and can be used along side them. In each chapter, we provide a related work section, in which, we compare and contrast our works with the related schemes.

2.8 Present Dissertation and Future Outlook

This chapter surveyed a variety of mobile sensing systems: Wireless Sensor Networks, Body Sensor Networks, Wearable Sensing, and Smartphone Sensing. We also discussed smartphone sensing paradigms: opportunistic and participatory, various sensors available in modern smartphones, and the applications of phone sensing systems to various domains like activity recognition, transport, entertainment, and user experience. We discussed in detail the application of smartphones to the social and behavioural sciences as this is the main focus of the current dissertation.

Energy is one of the main limiting factors of mobile phone applications. We have seen that improvements in mobile phone batteries have not matched those in other resources such as sensors, computing, and memory. Therefore, we need efficient techniques that can gather data from sensors and process them to draw inferences about the user. Once these techniques are in place many interesting applications can be built on phones providing great value to end-users and researchers.

This thesis is a step in this direction. Given the numerous applications of mobile phone sensing, especially in the social and behavioural sciences, we design three schemes to efficiently support social sensing applications on mobile phones: an adaptive sensing scheme (Chapters 3), a sensing offloading scheme (Chapter 4), and a computation offloading scheme (Chapter 5). To demonstrate the usefulness of smartphones in the conduct of social studies, we then present the design of three social psychological applications based on the services of the proposed schemes (Chapter 6).

3 Adaptive Sensor Sampling

3.1 Introduction

As discussed in the previous chapter modern smartphones are equipped with powerful sensors. In order to infer social context and to draw inferences about the behavioural aspects of the user, raw data from the sensors has to be captured continuously. However, continuous sensing leads to rapid depletion of the phone battery. It is reported, for example, in [WLA^{+}09] that the battery charge of a Nokia N95 [N95] smartphone lasts less than 5 hours when sensing data from the accelerometer, GPS, microphone, and Wi-Fi sensors using a predefined static (and aggressive) sampling rate. But if the sensors are sampled at a lower rate, then the system might not capture important events, resulting in lower accuracy. Therefore, there is a need to design adaptive mechanisms for querying the sensor data that achieve energy savings while maintaining the accuracy levels required for applications to be functional. Mobile phone operating systems lack a service that provides an adaptive sampling functionality. Therefore, there is a key opportunity to develop such a service for adding value to smartphone operating systems such as the Android platform, so that application developers can utilise it and need not implement these complex mechanisms but rather focus on their application functionality.

In this chapter we present a method to study and evaluate the energy-accuracy trade-offs of sampling rate control mechanisms for querying sensor data in continuous sensing mobile systems. We first present the method using various common function based rate control mechanisms (such as exponential, linear, etc.). We then present an adaptive sampling technique that achieves considerable energy savings while maintaining the required accuracy level through the use of *linear reward-inaction learning* [BH75, KLM96]. The adaptive sampling scheme adjusts the sampling rate of the sensors dynamically according to the context of the user measured in terms of events observed (interesting or not) and thereby achieves energy savings while maintaining accuracy, 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 also design and implement an API so that mobile applications can use the services of the adaptive sampling technique. Further, we design a rules framework that allows experiment designers (for example, social scientists) to write simple rules to program the behaviour of the sensor sampling. Adaptive sensing scheme is a step towards answering *Research Question 1 (How can we accurately capture raw data from the sensors in smartphones in an energy-efficient way?)* presented in Chapter 1.

Chapter outline. The remainder of the chapter is organised as follows: in Section 3.2 we describe our design method and present the learning-based adaptive sensing scheme. We present the rules framework in Section 3.3 and implementation details in Section 3.4. In Section 3.5 we present the evaluation of the proposed dynamic scheme along with a continuous sensing scheme and set of static functions with respect to energy-accuracy trade-offs. We present related work in Section 3.6, and finally, we present conclusions in Section 3.7.

3.2 Adaptive Sensor Sampling

In this section, we present the design of the adaptive sensing framework along with the learning based adaptive sensing scheme.

3.2.1 Design Method

In order to balance the energy-accuracy trade-offs of social sensing applications we present a design method and a set of functions that can be used to control the sampling rate of the sensors. We then present a learning-based technique to adapt the sensor sampling rate to the user's context, and show that this scheme performs better than a continuous sensing scheme and simple function based static schemes in balancing the energy-accuracy trade-offs.

Sensor Sampling

Sensor sampling is the process of sampling from the sensors of a smartphone. The process involves continuous *sense* and *sleep* cycles. In a sense cycle (or a sense window), data is queried from the sensor to infer events. In a sleep cycle (or a sleep window), the sensor is

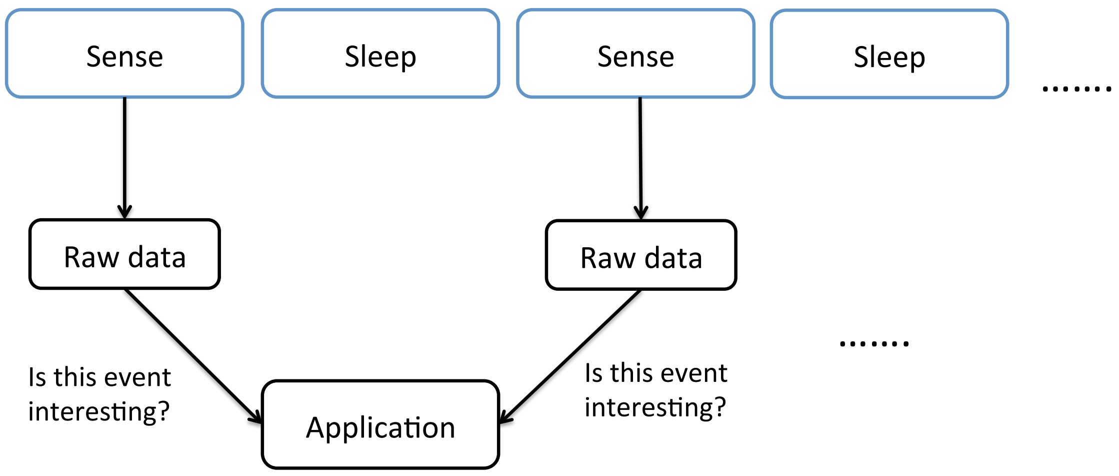


Figure 3.1: Sensor sampling process that we consider consists of consecutive sense and sleep windows. Data captured from each sense window is classified to interesting or uninteresting by the application requesting sensor data.

deactivated and no data is captured. *Sampling interval* is defined as sleep interval length between two consecutive sense cycles, and *sampling rate* as the number of sense cycles per unit time. We note that sensors also have a hardware-level sampling rate parameter, which is generally set by application based on its requirements. We do not consider hardware-level sampling rate in the design of adaptive sensing as it is application/classifier dependent. The sensor sampling process that we consider is as shown in the Figure 3.1.

Classification of Events

To support social sensing mobile phones should detect various social and behavioural events of the user. However, not all of these events are of interest to applications. For example, a speaker identification application might only be interested in audio samples that contain voice data, and may not be interested in audio samples that contain silence. Another example: consider an application like Google Latitude[1] that captures the user's location and updates his/her friends. When a user stays at a location there is no benefit in repeatedly capturing the same location information, but it will be enough just to detect a change in location. If a user changes location, then it is interesting to the application as it can update the user's friends view. Accordingly, we classify context events inferred from the raw sensor data into two classes *viz.*, *uninteresting* and *interesting*. An *uninteresting* event indicates that no interesting external event has occurred and the corresponding sensor can sleep (and need not sense) during this time. A *interesting* event is an event of interest to the application that has occurred in the user's environment that should not be missed by the sensor.

[1]http://www.google.com/mobile/latitude/

As mentioned earlier in this section, the sensor sampling process consists of continuous sense and sleep cycles. In each sense cycle, raw data from the sensor is captured and passed on to the application to check if it is interesting or not. The classification of events into interesting or uninteresting is entirely dependent on the application, which is requesting the sensor data, and is performed per sense cycle. The reason for the classification of sensor data is to exploit this knowledge for the adjustment of sensor sampling, which will be detailed later in the section. In this work we focus on context events that are represented by a stream of discrete states, such as streams of user activity like walking, sitting, conversing, and so on. The proposed method is also applicable to context information that is continuously changing such as temperature, as it can be discretely sampled.

Max and Min Sampling Intervals

As described, sensor sampling is a process of continuous sleep and sense cycles. The shorter the sleep cycles the higher the energy consumption (and the number of events detected). The longer the sleep cycles the lower the energy consumption, however, this may result in missing some interesting events. We define two parameters *viz.*, *minSamplingInterval* and *maxSamplingInterval*. The former is the minimum sleep interval between two successive sensor samplings and the latter is the maximum sleep interval. If the sensor sampling interval for a sensor is always set to *minSamplingInterval* (i.e., a constant rate sampling with sleep interval set to minSamplingInterval), then the accuracy of the classifiers will be high (due to aggressive data sampling). However, the energy expended will also be considerable. On the other hand, if the sampling interval is always set to *maxSamplingInterval*, then the energy consumption will be minimised, however, the accuracy will also decrease. Sense cycle, sensing window, or sampling window length is generally constant and is application dependent, for example, the length of an audio sample to be captured in each sensing cycle for noise detection is set to 5 seconds for speaker identification [RMM$^+$10b], or an accelerometer sample length of 5.12 seconds at 50Hz (for 256 samples) for activity detection [RDML05]. Thus, an important parameter that impacts energy and accuracy, and should be dynamically varied is the sleep duration between sensing cycles.

Back-off and Advance Functions

Intuitively, if no interesting events are observed, the sleep interval length between two consecutive sensor sampling cycles should increase, and if interesting events are observed, then the sleep interval length should decrease. Given this, we define two types of functions that vary the sleep interval duration.

Function type	**Back-off function**	**Advance function**
Linear	$2 \times x$	$x/2$
Quadratic	x^2	$\sqrt{x}$
Exponential	2^x	$\log_2 x$

Table 3.1: Back-off and advance functions.

- **Back-off function.** If no "interesting" events are observed (*i.e.*, missable events), then the sampling interval increases (*i.e.*, sampling rate decreases) from its current value to *maxSamplingInterval* based on a *back-off function*.
- **Advance function.** If a sensed event is classified as unmissable, then the sampling interval decreases (*i.e.*, sampling rate increases) from its current value to *minSamplingInterval* based on an *advance function*.

For example, let us assume that the current sampling interval for the microphone sensor in the user's phone is s. If no interesting events are observed (i.e., the classifier detects that none of the users is speaking), then the rate of sampling should be decreased to save battery. A way to do it is by increasing the sleep interval to $2 \times s$. If no interesting events are detected in the next iteration, then we further increase the interval and so on until it reaches a maximum value (*maxSamplingInterval*). Even though this saves energy (as the function increases the sleep interval), it might miss some interesting events during the sleep time. Similarly, if the system detects an interesting event, then the sampling rate is increased. A way to do it is by decreasing the sleep interval to $s/2$. Combining event classification, max/min sleep intervals, and advance and back-off functions, the adaptive sensing design works in the following way.

Once a sensor is sampled the captured data is processed and classified by the application-level classifiers as an interesting or uninteresting event. If classified as uninteresting, then the sleep interval increases from its current value based on a back-off function, such as sleep interval = 2 × sleep interval. If the event is classified as interesting, then the sleep interval decreases from its current value based on an advance function, such as sleep interval = sleep interval / 2. The sleep interval is bounded by max and min intervals to avoid too conservative or too aggressive sampling of the sensors. The choice of the advance and back-off functions and of the *minSamplingInterval* and *maxSamplingInterval* parameters plays a crucial rule in the energy-accuracy trade-offs of the various context inference components. Examples of the back-off and advance functions (also used in the

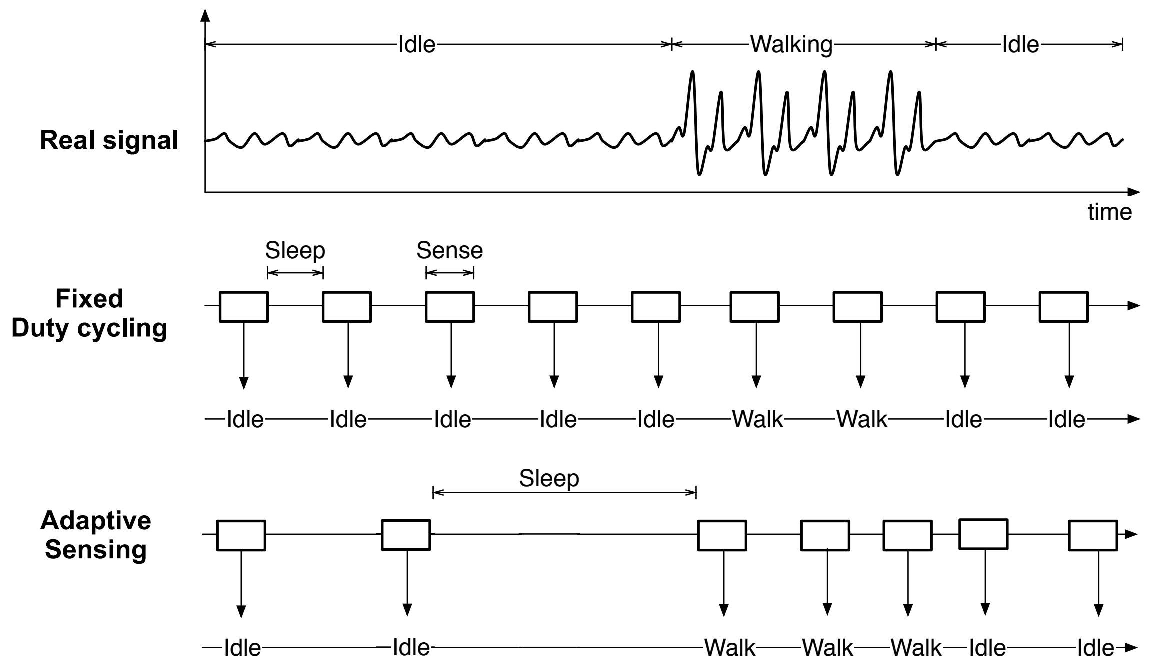


Figure 3.2: Example scenario: Fixed duty cycling vs adaptive sensing for the accelerometer sensor.

evaluation in Section 3.5) are given in Table 3.1. The back-off function is invoked when there are no interesting events observed and the advance function is invoked otherwise.

Figure 3.2 shows an example scenario of the accelerometer sensor using a fixed duty cycling sampling and the proposed adaptive sensing. In this example a *moving event* (walking) is considered interesting and a *stationary event* (idle) is considered uninteresting. Since the fixed duty cycling scheme always samples at the same rate, it may waste energy by sampling even when no interesting events are observed. Further, it may also not capture all interesting events as it does not decrease the interval on detecting *moving* events. The adaptive sensing scheme adjusts the sampling rate according to the user's context. It decreases the sampling rate on detecting uninteresting events and increases the sampling rate on detecting interesting events. We will present the energy savings and the energy-accuracy trade-offs of the fixed and adaptive schemes using benchmark tests in Section 3.5.

3.2.2 Learning based Adaptive Sensing

In the previous section we used various static functions to describe the advance and back-off functions. However, in order to fully utilise the adaptive sensing design, the behaviour of the advance and back-off functions should change according to the user's context, i.e., the rate of increase or decrease of the sampling interval should be dynamic. For example, the rate of decrease on detecting an interesting event should not always be the same and should depend on the recent history (in terms of interesting and uninteresting events) of the user. Therefore, in this subsection we present the design of a machine learning technique that dynamically adapts the sampling rate of the sensors according to the user's context, achieving improved energy savings without compromising the accuracy of mobile phone applications.

Learning based Adaptation

We design a learning technique based on the theory of learning automata [NT89] to control the sampling rate of the sensors. In particular, the algorithm is based on the *linear reward-inaction* scheme [BH75, KLM96] [2]. 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 or not to sense from a sensor in each sensing window is based on a probability value, which we call *probability of sensing*. In cases where the scheme decides to sense the sensing action results in either *success* or *failure*, which are defined 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 call this a *success*. Similarly, when data sensed from a sensor corresponds to a missable event (for example, audio data from the microphone sensor contains no audible data but only silence) then we call this a *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 a *success*, otherwise a *failure*. The probability of sensing from a sensor is dynamically adjusted according to the 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 the sensor.

If the sensing action a_i results in an unmissable event then it means that the system has detected an interesting event (i.e., a *success*, e.g., an audio sample containing voice data), so the probability is increased (or advanced). The probability value (p_i) is calculated according to the following formula:

$$p_i = \tilde{p_i} + \alpha_I(1 - \tilde{p_i}),\ where\ 0 < \alpha_I < 1 \tag{3.1}$$

$\tilde{p_i}$ is the probability of sensing calculated in the previous step and α_I (alpha increase) is a constant factor that sets the rate at which the probability increases. When $\tilde{p_i}$ is low, then $(1 - \tilde{p_i})$ is high and therefore p_i increases more quickly. When $\tilde{p_i}$ is high, then $(1 - \tilde{p_i})$ is small and therefore p_i increases at a slower rate. The intuition behind this model is that the probability increases more quickly when it is at the low end of the range $[0, 1]$, and more slowly as it reaches the higher end. To explain further, if the probability is low,

[2] Alternative techniques include information theoretical approaches such as compressed sensing [Don06], which are orthogonal with respect to our application-level approach.

and interesting events are observed, then it has to increase more quickly to increase the sampling rate. If the probability is high, and interesting events are observed, then it can grow at a slower rate as it is already high. This formula is used as an advance function when an interesting event is detected.

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 (or backed-off). The new probability value is calculated according to the following formula:

$$p_i = \tilde{p}_i - \alpha_D \tilde{p}_i, \ \textit{where} \ 0 < \alpha_D < 1 \tag{3.2}$$

$\tilde{p}_i$ is the previous probability of sensing and α_D (alpha decrease) is a constant factor that controls the rate at which the probability decreases. When $\tilde{p}_i$ is low then the decrease of the probability will be slower, and when $\tilde{p}_i$ is high then the decrease of the probability value is faster. Similar to the previous explanation, the intuition is that when $\tilde{p}_i$ is high and an uninteresting event is detected, then it has to decrease more quickly to save energy as the sensing does not lead to event detection. If the $\tilde{p}_i$ value is low and an uninteresting event is detected, then the decrease rate could be slower as the probability is already low. This formula is used as a back-off function when an uninteresting event is detected. By adopting these mechanisms, the sampling rate adapts to the context of the user. The flow diagram of the learning based adaptive sensing scheme is shown in Figure 3.3.

In the initialisation phase, since we do not have the previous context of the user, we set the probability value to 0.99, so that the sensing is performed almost continuously at the start and then it subsequently adapts according to the sensed user's context. If we set the probability to a lower value, since we do not have the user's previous context, it might result in missing some interesting events, therefore, we chose to set the initial probability to a high value. If the probability value falls too low, then it might take several iterations of detecting interesting events to reach a higher probability value, and this might result in missing interesting events. Since accuracy is an important factor that should be maintained for social sensing applications to be functional, we limit the lower bound of the probability to 0.1, i.e., converges to 10% duty cycling. We evaluate this learning technique and compare its performance with that of various static functions, a continuous sensing scheme, and a fixed duty cycling scheme in Section 3.5

Assumptions

In order to apply the learning based adaptive sensing scheme, we assume that the following requirements hold.

- The application should be able to classify events to interesting or uninteresting according to its requirements. If an application is unable to do this, then the proposed scheme can not be applied.

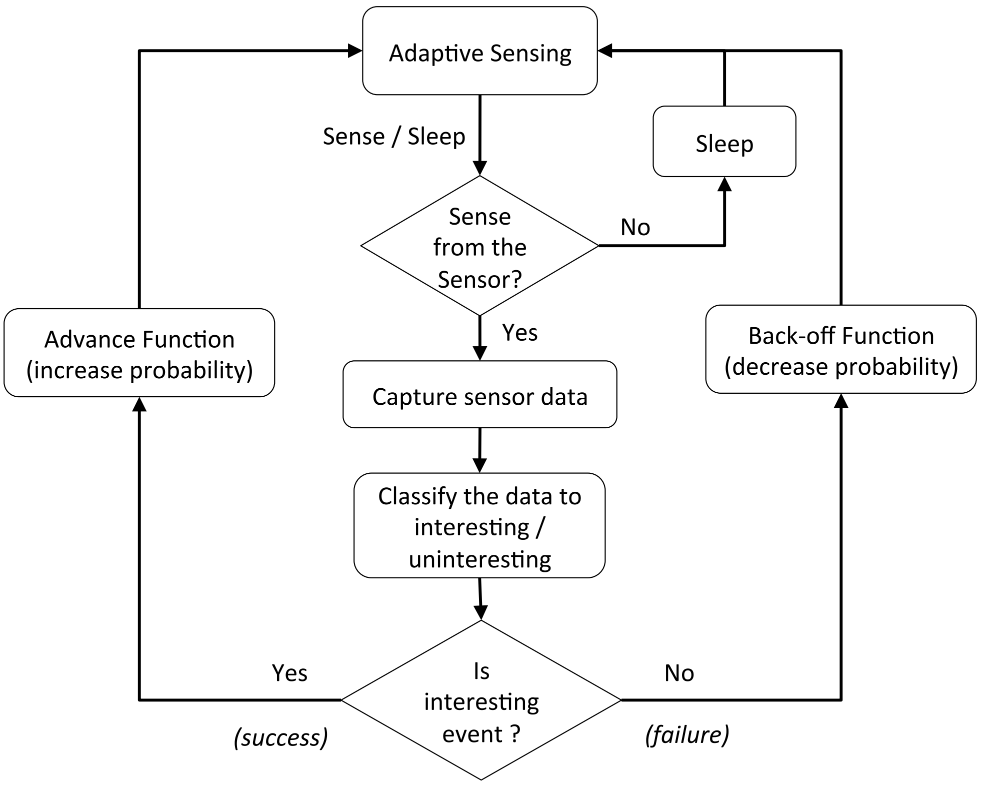

Adaptive Sensing
Sense / Sleep
Sleep
Sense from the Sensor?
No
Yes
Advance Function (increase probability)
Back-off Function (decrease probability)
Capture sensor data
Classify the data to interesting / uninteresting
Is interesting event ?
Yes
(success)
No
(failure)

Figure 3.3: Flow diagram of the learning based adaptive sensing scheme.

- If an interesting event occurs in a large sequential stream of uninteresting events, then there is a high chance that the scheme may miss the event, as it needs a sequence of interesting events to increase the probability of sensing. The proposed scheme, therefore, is not useful to capture events that happen rarely.
- The proposed scheme is applicable only to "pull sensors", and "push sensors" do not need a duty cycling mechanism as described in Section 2.6.

3.2.3 Adaptive Sensing API

Modern mobile phone operating systems such as the Android system provide APIs for sampling the sensors. However, they lack a service to dynamically adjust the sensor sampling interval. We therefore designed and implemented APIs in order for mobile sensing applications to utilise the services of the proposed adaptive sampling scheme. The design works in the following way:

- The sensor monitors (or sensor trackers or sensor threads that are responsible for capturing data from the sensors) should first register a listener with the adaptive sampling service providing details about the sensor type and sensor-specific categorisation of interesting/uninteresting events.

- Whenever a sensor monitor captures data from the sensor, it should be sent to the adaptive sampling service so that the service can adjust the probability of sensing (described in the previous subsection).
- The adaptive sensing component then updates the sampling interval according to the probability of sensing and notifies the sensor monitor asynchronously.

A detailed description of the APIs and sample code to show the usage of the APIs are provided in Appendix A.

3.3 Rules Framework

The users of social sensing systems may not have necessary programming experience to customise the sensing process, therefore, we designed a rules framework to support the writing of rules to trigger sensing actions based on context events. Rules can be written to capture data from a sensor on detecting a specific context event; for example, capture data from the microphone only when the user is at home, or capture data from the Bluetooth only when the user is at the office. The rules framework can also be used to trigger an expensive (in terms of power) sensor based on an inexpensive sensor to save energy such as the accelerometer based triggering of the GPS.

The adaptive sensing framework described in the previous section captures data from several sensors. This data is then logged into two declarative databases (*Knowledge Base* and *Action Base*). The data from each of the sensors is logged to the *Knowledge Base*, a repository of all the information extracted from the on-board sensors of the phones. The framework is based on a declarative specification (using first-order logic predicates) of:

- *facts*, i.e., the knowledge extracted by the sensors about user behaviour (such as the user's emotions) and his/her environment (such as the identity of the people in conversation with the user).
- *actions*, i.e., the set of sensing activities that the sensors have to perform, such as recording voices (if any) for 10 seconds each minute when the user is at home or extracting the current activity every 2 minutes when the user is moving.

The sensing actions are periodically generated by means of the inference engine and a user-defined set of rules (a default set is provided). The actions that have to be executed by the system are stored in the *Action Base*. Users of the framework can define sensing tasks and rules that are interpreted by the inference engine in order to dynamically adapt the sensing actions performed by the system. In the remainder of this section we discuss details of the declarative databases and inference engine, while presenting several example rules.

3.3.1 Action and Knowledge Base

The *Knowledge Base* repository stores the current facts that are inferred from the raw data generated by the various sensors. Sensor trackers or sensor monitors capture data from the sensors in the phone and log them into the *Knowledge Base*, which are in turn used by the inference engine to generate actions. As the framework is aimed at mobile sensing systems the *Knowledge Base* is designed to load in "memory" only the most recent facts to reduce the application footprint. The format of facts is as follows:

```
fact(<fact_name>, <value>)
```

The corresponding timestamps of these facts are also stored. Actions are also treated as facts, but with an extra identifier which is of the form:

```
fact('action', <action_name>, <value>)
```

Some examples are:

```
fact(Activity, 1)
fact('action', 'ActivitySamplingInterval', 10)
```

The former indicates that the user is currently moving, and the latter means that the sampling interval of the accelerometer should be set to 10 seconds.

If users of the framework add rules to control the duty cycling of the phone's sensors, then these rules override the adaptive sensing functionality, *i.e.*, the sampling of sensors for which a sampling rule exists will not be controlled by the adaptive sensing, but will be performed as per the specified rule. For example, if a rule is added to sample from a sensor at a fixed rate then it overrides the adaptive sensing. Therefore the duty cycling of sensors should only be used if the users of the framework do not wish to use adaptive sensing for these sensors.

3.3.2 Inference Engine

The framework is based on a set of *adaptation rules* that enables the adaptation of the behaviour of the system at run-time by monitoring the current activity, co-location with other people, and location of the person carrying the mobile phone. The adaptation rules are used to modify the sampling behaviour of the system according to the observed status of the user (e.g., whether a person is moving), and his/her surroundings (e.g., if there are other people around, if they are currently talking). The framework reduces energy consumption by reducing data sampling and information processing by using triggers such as the accelerometer based triggering of the GPS sensor.

```
1. set_location_sampling_interval
2.  foreach
3.    facts.fact($factName, $value)
4.    check $factName == 'Activity'
5.    facts.fact($actionName, $currentInterval)
6.    check $actionName == 'LocationInterval'
7.    $interval = update($value, $currentInterval)
8.  assert
9.     facts.fact('action', 'LocationInterval', $interval)
```

Figure 3.4: An example rule to set the sampling rate of the GPS sensor based on the accelerometer sensor.

```
def update(value, currentInterval):
 if value == 1:
   samplingInterval = 10 # seconds
 elif value == 0:
   samplingInterval = 3600 # a large value
 return samplingInterval
```

Figure 3.5: A function to update sampling interval.

An example of a rule used is given in Figure 3.4. The rule updates the value of the location sampling interval in accordance with data from the accelerometer sensor. The rule retrieves the fact `Activity` (lines #3 and #4) and the current location sampling interval `LocationInterval` from the *Knowledge Base* and then executes an action (line #9) to update the sampling interval based on a function (`update()`, Figure 3.5). The idea is to provide a simple interface to add rules in order to change the behaviour of the system. Another example is to write a rule to capture the data from the microphone when the user is at a specific location (shown in Figure 3.6). In this case, when the user is at home the `$value` variable (retrieved in line #3) holds an integer value representing this state.

```
set_mic_sampling_interval
 foreach
   facts.fact($factName, $value)
   check $factName == 'Location'
   facts.fact($actionName, $currentInterval)
   check $actionName == 'MicInterval'
   $interval = update($value, $currentInterval)
 assert
    facts.fact('action', 'MicInterval', $interval)
```

Figure 3.6: An example rule to set the sampling rate of the microphone sensor based on the user's location.

3.4 Implementation

The adaptive sensing scheme has been implemented on the Symbian S60 platform based on Python for S60 [PYS] and the Android operating system 2.1. We implemented *sensor monitors* to capture data from the accelerometer, Bluetooth, microphone, and GPS sensors. For the Symbian S60 platform we used the accelerometer sensor API provided by PyS60 platform to access the X, Y, Z axes data of the accelerometer sensor, and the *lightblue* [LIG] module to perform Bluetooth discovery operations. We used the *positioning* module of PyS60 to capture the user's location information. The monitor tries to extract valuable information even if the GPS data is not complete. An example of an incomplete position is one that contains data about the satellites used to obtain a GPS fix but no latitude or longitude data. This can at least be used to infer whether the user is indoors or outdoors.

Android Bluetooth APIs were used to discover Bluetooth devices in 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 device found to retrieve its Bluetooth name. The Android *SensorManager Service* is used to access the X, Y, Z axes of the accelerometer sensor data. This process is asynchronous too and involves registering a listener (*SensorEventListener*) for capturing accelerometer data. The Android SensorManager API provides various speeds at which data is sensed from the accelerometer, and we use the *SENSOR_DELAY_FASTEST* setting to capture accelerometer data as fast as possible. The location information is obtained through the Android Location Manager [ALM].

The rules framework is based on Pyke [PYK], a knowledge-based inference engine. It takes a set of facts as inputs and derives additional facts through forward chaining rules. It can also be used to prove goals using backward chaining rules. However, these are not necessary in our system and were removed when we adapted Pyke for our platform in order to reduce the memory footprint. We have also provided a set of adaptation rules along with the framework which drive the behaviour of some sensors. The inference engine is periodically invoked to process facts and generate actions.

3.5 Evaluation

In this section we present the evaluation of the learning based adaptive sampling scheme with respect to accuracy and energy using real traces collected by participants carrying mobile phones.

3.5.1 Empirical Datasets

We collected the dataset for benchmarking the various sensor sampling schemes from users carrying mobile phones. We gathered a total of 255 hours of raw accelerometer sensor data (i.e., X, Y, Z coordinates of 3-axis accelerometer), 230 hours of Bluetooth data (i.e., Bluetooth identifiers), and 213 hours of microphone data (i.e., audio recordings), with 10 users carrying a Samsung Galaxy S phone (running Google Android 2.1 Platform[3]) or a Nokia 6210 Navigator phone (running Symbian S60 platform[4]). The participants were students and staff of the Computer Laboratory, University of Cambridge. The data was collected during weekdays. The participants were not required to interact with the sensor data collection application as it was designed as a passive data collection tool. The data was not collected from all the participants during the same days but was collected over a period of few weeks based on their availability. The sampling of the accelerometer sensor and the Bluetooth sensor was performed continuously with a sleep interval of 0.5 seconds and 1 second, respectively. Audio samples of length 5 seconds were recorded from the microphone sensor with a sleep interval of 1 second between consecutive audio recordings.

3.5.2 Classifiers

The adaptive sensing scheme requires the data from the sensors to be classified as interesting or uninteresting events in order to dynamically adjust their sampling intervals. Therefore, after the dataset was collected, we classified the raw data in the sensor traces into high level events. For example, [x, y, z] vector data from the accelerometer sensor

[3]http://developer.android.com/about/versions/android-2.1.html

[4]http://www.developer.nokia.com/Devices/Symbian

can be classified as *moving* or *idle* states. We then mapped these high level events to interesting or uninteresting events. The classification process was executed offline to generate the event traces from the raw sensor traces. In particular, we used the following classifiers:

Movement Classifier

The movement classifier classifies raw data from the accelerometer sensor to *moving* or *stationary* events. Raw data from the accelerometer contains a list of [x, y, z] vectors. The classification procedure is as follows:

- Calculate the magnitude of the acceleration of each of the vectors using the formula: for a vector i, magnitude of acceleration $m_i = \sqrt{x^2 + y^2 + z^2}$.
- Divide the list of vectors into three subsets: first one-third, second one-third, and last one-third. For each of these sets calculate the standard deviation of the magnitude of acceleration values.
- For each set, if the standard deviation is greater than a certain threshold value (which is determined using labelled ground truth data), then the user is considered to be moving. The result is the event given by at least two of the three sets. Predefined threshold based approach has been used by some works such as [FF00] and [TRB+11].

Conversation Recognition

The conversation recognition classifier classifies a raw audio file as *silence* or *sound.* The classifier was implemented using the Hidden Markov Model Toolkit (HTK) [HTK]. Two Gaussian Mixture Models representing speech and silence models were trained, and then each raw audio file was parameterised and compared with the conversation and silence models. The model with highest likelihood of match is assigned as the model of the recorded audio file. More details of this classifier will be presented in Chapter 6.

Co-location Change Detection

The co-location change classifier uses the data from the Bluetooth sensor to detect whether the co-location of the user (i.e., the Bluetooth device addresses captured by the user's phone) has changed with respect to the data from the previous sensor sampling. It is a simple classifier, and compares the Bluetooth scanning results of two consecutive sensor samplings to detect whether the user is currently co-located with the same or a different set of users. The Bluetooth discoverability on all the phones carried by the users was set to ON during the entire data collection process.

3.5.3 Methodology

In this subsection, we describe our evaluation methodology.

Categorisation of Events

As discussed in Section 3.2, we classified the raw data in the sensor traces into events, which can be of two types, *viz.*, "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. This is because audio data with sound can be used by the classifiers for further processing like speaker identification [LBBP+11] or stress detection [LRC+12], but silence data may not provide much information. For the Bluetooth sensor traces an unmissable event corresponds to a change in the number of co-located users, whereas a missable event indicates no change. If there is no change in the user's co-location state, then the mobile system/application already has this knowledge, so there is no need to spend energy in detecting a known event. 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 evaluated the performance of the adaptive scheme with respect to the metrics: *accuracy* and *energy*. The accuracy is measured in terms of the percentage of interesting events detected. We do not consider uninteresting events as they may either provide information that is redundant or not of much value to mobile applications.

We measure the energy consumption using the Nokia Energy Profiler (NEP) [NEP]. NEP is a stand-alone test and measurement application for the Symbian S60 platform provided by Nokia and it provides a way of measuring the power consumption of the mobile phone at fine-grained time intervals. It can be used to estimate energy consumption, cumulative energy consumption, as well as battery voltage and current, etc. The tool's documentation mentions that when an application is in use, the Nokia Energy Profiler can be run in the background to profile power consumption. It is a popular energy measurement tool for measuring energy consumption on Nokia mobile devices [VRC13] and many research works such as [PFS+09, BBV09, XSK+10, WLA+09] have used it to measure and estimate the power consumption of the mobile phone.

The Nokia Energy Profiler uses a power model to estimate the energy consumption of various operations on the phone. Power models are generally built based on the direct measurements of the power consumption of various components of the phone [ZTQ+10].

Like most of the energy measurement tools based on power models, NEP is also prone to small inaccuracies, related to factors such as temperature, total recharge cycles of battery etc. However, we are interested in the relative differences in the energy consumption but not the absolute energy measurement values. Therefore, this tool should provide us an estimate of the relative differences between schemes as we are interested only in the energy difference across schemes. The energy consumption of a sensing task is computed as follows: we first measure the baseline power consumption of the mobile phone, and then we activate the sensing task and measure the energy consumption. The difference between these energy consumption values is calculated as energy of the task. We repeat this procedure for over 100 iterations and finally we calculate the average of these values to determine the average energy consumption of the task.

Power measurement values of the sensing tasks are affected by many factors such as operating system, CPU frequency, phone model, and background processes. In the evaluation the energy consumption of a scheme with respect to a sensor is estimated from the length of time the sensor was active and the measured power value. Thus differences in the power measurement values across devices and operating systems should not affect our findings, as we evaluate the relative difference between the schemes. For example, if a scheme activates a sensor for x duration of time, then the energy is estimated by multiplying this by the measured average power value. If another scheme activates the sensor for $2x$ length of time achieving the same level of accuracy, then we can say that the former scheme saves 50% energy more than the latter.

Tuning of the Adaptive Scheme

We first fine-tuned the parameters of the adaptive sampling scheme: since the α values might vary according to the requirements of the application, we explored the performance of the learning technique for each sensor for the entire parameter space of α_I and α_D values (explained in Section 3.2) and then selected the appropriate values for them. We measured the performance of the sampling schemes presented in the previous sections, based on the percentage of total interesting context events (as described earlier in this subsection) detected. The number of context events generated and its distribution is generally dependent on the behaviour of the user. For example, the accelerometer sensor in a user's phone, who is not physically active during office hours, might generate a large number of "user stationary" events during working hours. Similarly, the phone of a user who is physically active during office hours might sense "user moving" events more often. Another example is that the behaviour of the user in terms of her speech patterns impacts the distribution of "user speaking" events detected by the phone's microphone. Therefore, the α values selected by this process should be general enough to provide the same level of performance in the cases with a similar user behaviour as this might result in similar distribution of context events.

Techniques used for Comparison

To quantify the advantages of adaptive sensor sampling, we compared its performance with a continuous sensing scheme, a fixed 50% duty cycling scheme, and various static back-off and advance functions. The back-off and advance functions used in the evaluation are given in Table 3.1 (Section 3.2).

3.5.4 Results

Bluetooth Sensor

α_I and α_D variation. To optimise the learning based technique for each of the sensors, we first measured its performance in terms of accuracy and energy by varying the α_I and α_D values. Figures 3.7 and 3.8 show the variation of both the α values with respect to the performance of the learning based technique for the Bluetooth sensor in terms of accuracy and energy, respectively. From these plots we can observe that the technique is more accurate for higher values of α_I and lower values of α_D. This is because for higher α_I values the probability of sensing increases more rapidly and for lower α_D values the probability decreases slowly, i.e., the probability increases faster on detecting interesting events, but decreases slowly on detecting uninteresting events, which, on average, results in higher probability of sensing values. The learning technique decides whether to sense or not in each sensing window, and in its most aggressive form (a high α_I and a low α_D value) it resembles a continuous sensing scheme. However, the main strength of the technique is its adaptiveness, which we demonstrate further. Using these results, the α_I and α_D values can be selected according to the energy and accuracy requirements of the application. To achieve high accuracy, values for α_I and α_D parameters based on the results could be 0.9 and 0.1, respectively.

Comparison with other sampling techniques. To understand the energy savings and adaptiveness of the learning based scheme we also compared it with a continuous sensing scheme and a 50% duty cycling scheme. Figures 3.9 and 3.10 show the performance of the adaptive, continuous sensing, and 50% duty cycling schemes with respect to accuracy and energy for the Bluetooth sensor. We can observe that, to achieve almost the same level of accuracy, the learning scheme consumes 50% less energy than the continuous sensing scheme. With respect to the 50% duty cycling scheme, the learning scheme consumes 10% less energy and achieves 40% greater accuracy. This is an important result considering that social sensing applications may require high accuracy to accurately model the user's behaviour, and users may not be willing to participate in the experiments if they have to frequently recharge their phones. The evaluation results show that the proposed learning based adaptive scheme satisfies both these requirements. Further, we also compared the performance of the adaptive scheme with that of various static functions described in

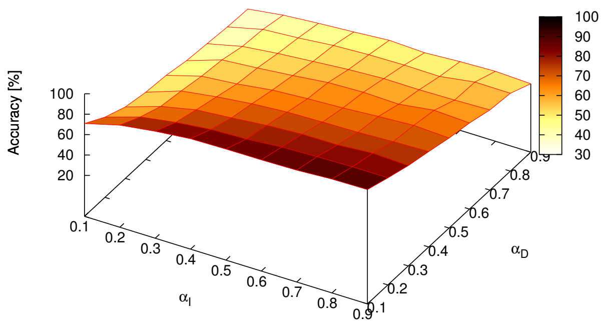


Figure 3.7: Accuracy vs alpha increase and alpha decrease for Bluetooth sensor.

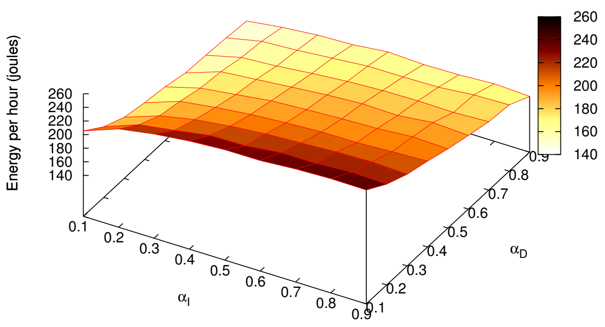


Figure 3.8: Energy consumption per hour vs alpha values for Bluetooth sensor.

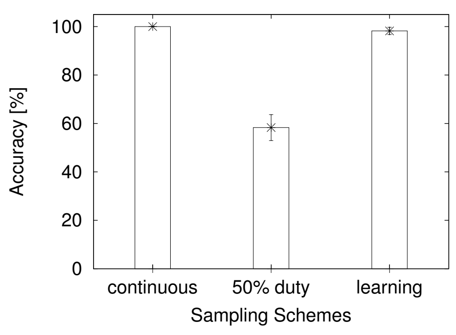


Figure 3.9: Accuracy of continuous, 50% duty cycling, and learning schemes for Bluetooth sensor.

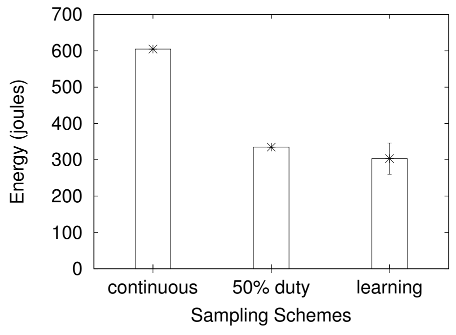


Figure 3.10: Energy per hour of continuous, 50% duty cycling, and learning schemes for Bluetooth sensor.

Section 3.2 (Table 3.1). Figures 3.11 and 3.12 show the performance of combinations of static functions and the learning technique. The x-axis labels in the graph are of the format *advance function – back-off function*. Even though the energy savings were higher for the static functions, the accuracy they achieved was very low: in consequence they may not be able to capture all the required events to accurately model the behaviour of the user. The static functions do not adapt to the context: they react in the same way to a single interesting event or a continuous stream of context events, achieving low accuracy. From the results it can be inferred that the combination that performs best and closest to the learning scheme is *exponential-linear*, as this exponentially decreases the sampling interval on capturing an interesting event and linearly increases the sampling interval on capturing an uninteresting event, i.e., the sampling rate increases at a high rate and decreases at a low rate. However, it is still 21% less accurate than the learning based scheme. We note that since the confidence intervals of these static combinations overlap, it is difficult to infer with high confidence about the performance differences among these schemes and the best performing scheme out of all the combinations.

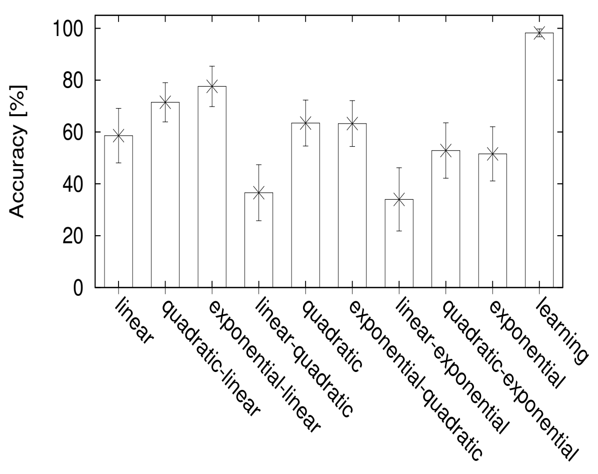


Figure 3.11: Accuracy of static function based schemes for Bluetooth sensor.

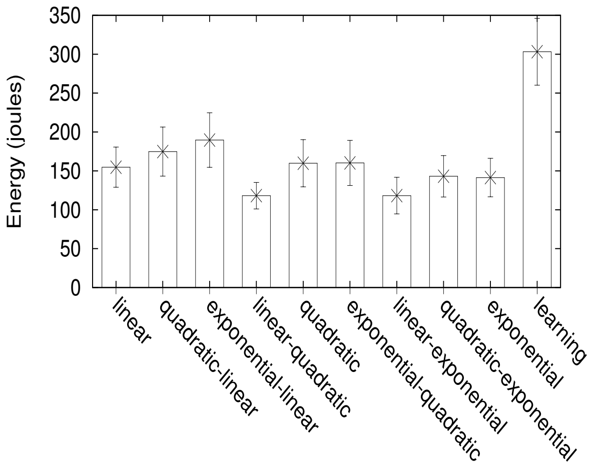


Figure 3.12: Energy per hour of static function based schemes for Bluetooth sensor.

Max/Min sleep interval variation. The maximum sleep interval (described in Section 3.2) value limits the amount of sleep time between two consecutive sensor sampling windows, and it affects the accuracy and energy of the adaptive learning scheme. To understand this we studied its effect on two combinations of alpha values: $\alpha_I = 0.9$, $\alpha_D = 0.1$ (aggressive sampling) and $\alpha_I = 0.1$, $\alpha_D = 0.9$ (conservative sampling). We set the minimum sampling interval to a value close to zero (2ms), so that the schemes sample almost continuously when sampling at the maximum rate. Figures 3.13 and 3.14 show the effect of increasing the maximum sleep interval on the accuracy and energy, respectively, of the adaptive scheme. As expected, as the sleep interval increases, the energy and accuracy of the scheme decrease. However, the rate of decrease is slower for the aggressive sampling combination of α values ($\alpha_I = 0.9$, $\alpha_D = 0.1$) than the conservative sampling combination ($\alpha_I = 0.1$, $\alpha_D = 0.9$). This is because in the former case the scheme reacts to interesting events by rapidly increasing the probability of sensing, and to the uninteresting events by slowly decreasing the probability of sensing, which results in higher average probability of sensing. Due to this, the scheme senses more often and does not frequently reach the maximum sleep interval value. Therefore, the effect of increasing the max interval value is lesser on this combination than on the other.

Accelerometer Sensor

In this subsection we present the evaluation of the adaptive scheme with respect to the accelerometer sensor. Figures 3.15 and 3.16 show the accuracy and energy consumption of the learning technique for different alpha values. We observe similar results to the Bluetooth sensor, i.e., accuracy is greater for higher α_I and lower α_D values. In addition, we observe that the variation of the energy values for accelerometer is not as high as that of Bluetooth, as the cost of sensing from the accelerometer is much lower than is the case with the latter. We now compare the performance of the adaptive scheme for high α_I

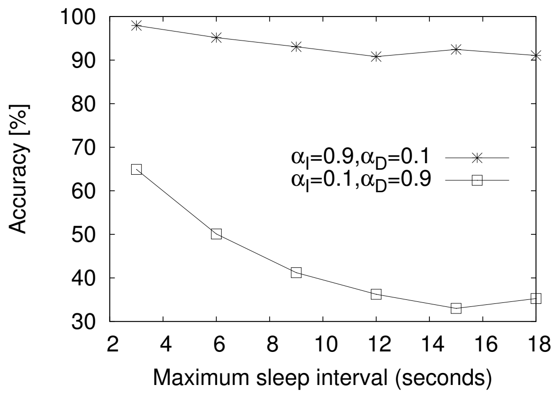


Figure 3.13: Maximum sleep interval vs accuracy of the learning scheme for two different α_I, α_D combinations.

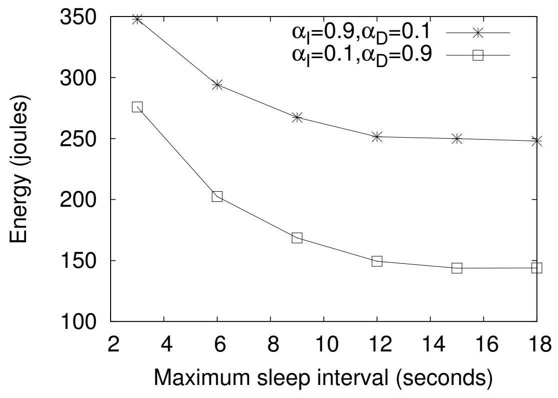


Figure 3.14: Maximum sleep interval vs energy per hour of the learning scheme for two different α_I, α_D combinations.

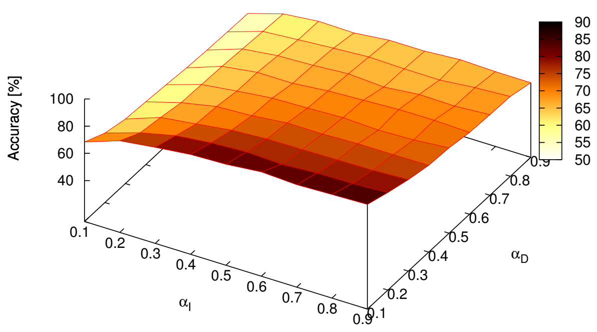


Figure 3.15: Accuracy vs alpha increase and alpha decrease for accelerometer sensor.

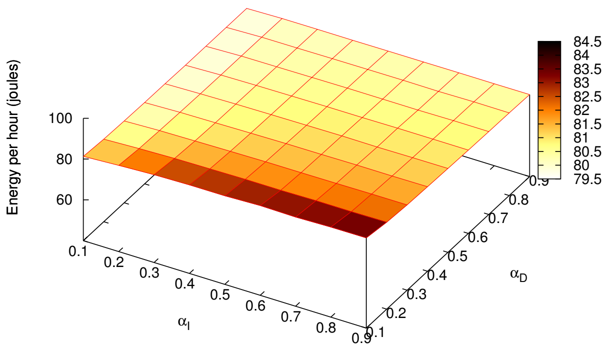


Figure 3.16: Energy consumption per hour vs alpha values for accelerometer sensor.

and low α_D values with continuous sensing and 50% duty cycling schemes, to determine the energy savings of the adaptive scheme. Figures 3.17 and 3.18 show the accuracy and energy results of the sensing schemes. We can observe that both learning and continuous schemes achieve close to 100% accuracy. However, the energy consumption of the adaptive scheme is 42% less than that of continuous sensing. With respect to the 50% duty cycling scheme, the learning scheme consumes 12% less energy and achieves 33% higher accuracy. We also measured the accuracy achieved by the static functions: these achieve low accuracy because they are static (Figures 3.19 and 3.20). Similar to the Bluetooth results, *exponential-linear* is the most accurate static combination, but it remains much less than the learning scheme. We note that the energy values of all the static combinations are very close because of the low power consumption of the accelerometer sensor, i.e., large differences in accuracy translate to small differences in energy.

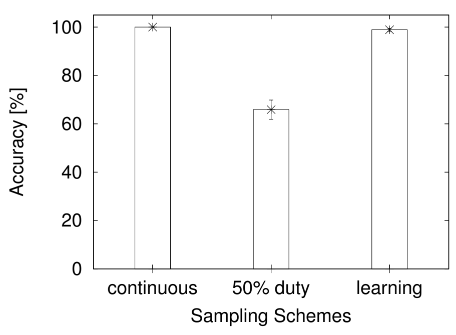


Figure 3.17: Accuracy of continuous, 50% duty cycling, and adaptive schemes for accelerometer sensor.

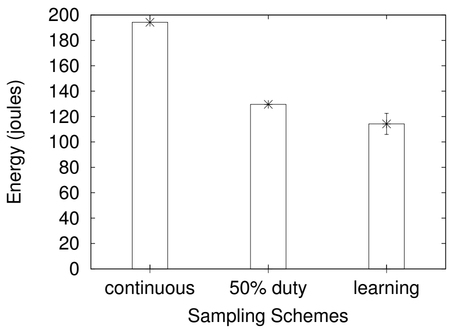


Figure 3.18: Energy per hour of continuous, 50% duty cycling, and adaptive schemes for accelerometer sensor.

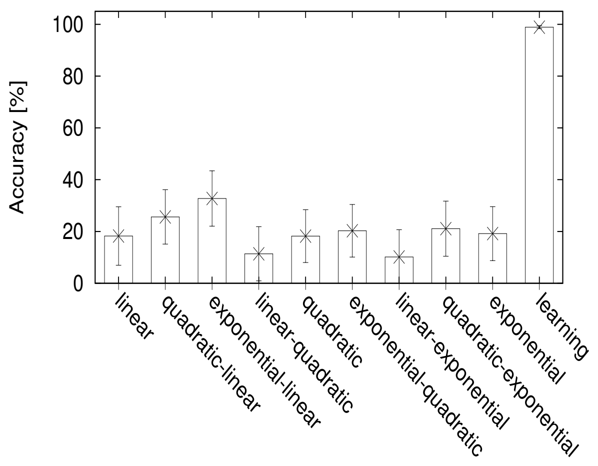


Figure 3.19: Accuracy of static function based schemes for accelerometer sensor.

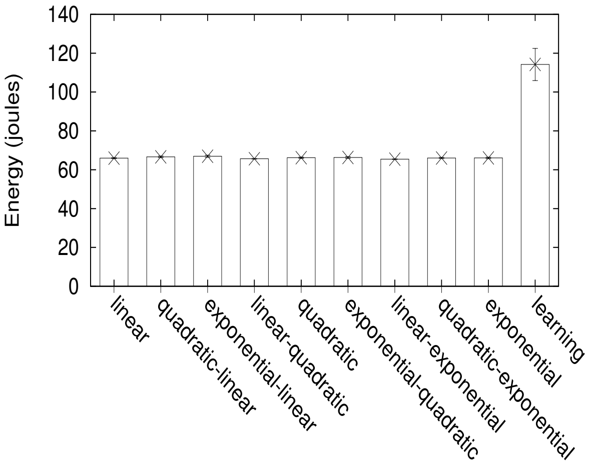


Figure 3.20: Energy per hour of static function based schemes for accelerometer sensor.

Microphone Sensor

Finally, we measured the performance of the adaptive, continuous, 50% duty cycling, and static functions for the microphone sensor. Figures 3.21 and 3.22 show the performance of the adaptive scheme for varying alpha values for the microphone sensor. As expected, the adaptive scheme is more accurate for higher α_I values and lower α_D values. Figures 3.23 and 3.24 show the accuracy and energy results for the continuous sensing, 50% duty cycling, and adaptive sensing schemes. The results show that while achieving accuracy results close to 100% the learning scheme consumes 43% less energy than the continuous sensing scheme. With respect to the 50% duty cycling scheme, the learning scheme consumes 7% more energy due to the intensity of the audio processing task. However, it achieves 30% more accuracy than the duty cycling scheme. We also evaluated the static functions for the microphone sensor and observed similar behaviour to that of the other sensors, i.e., the accuracy achieved was very low.

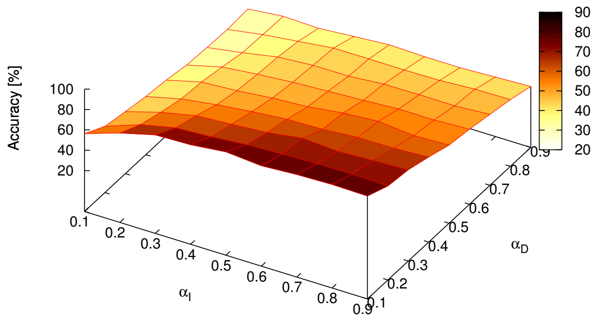


Figure 3.21: Accuracy vs alpha increase and alpha decrease for microphone sensor.

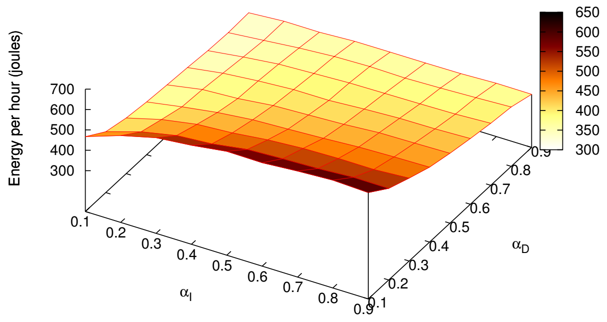


Figure 3.22: Energy consumption per hour vs alpha values for microphone sensor.

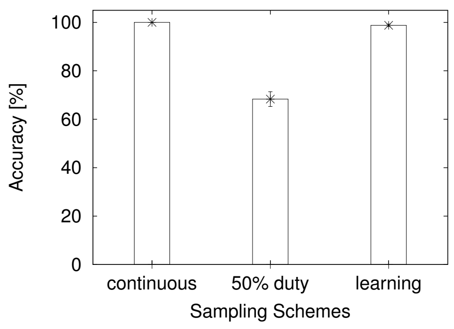


Figure 3.23: Accuracy of continuous, 50% duty cycling, and adaptive schemes for microphone sensor.

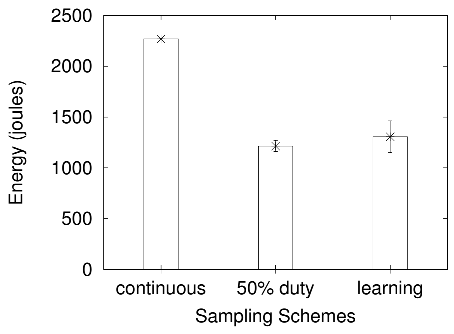


Figure 3.24: Energy per hour of continuous, 50% duty cycling, and adaptive schemes for microphone sensor.

The results presented in this section demonstrate that the proposed learning scheme is more energy-preserving than the continuous sensing scheme while achieving similar accuracy for all the sensors. The energy savings achieved by the learning based adaptive scheme are 50%, 42%, and 43% higher compared with the continuous scheme with respect to Bluetooth, accelerometer, and microphone sensors, respectively. Compared with the fixed 50% duty cycling scheme, the learning scheme achieved much higher accuracy for a comparable (microphone) or better energy savings (Bluetooth, accelerometer). The results also showed that the static function-based schemes achieve too low an accuracy to be useful for accurate social sensing. Further, the α_I and α_D parameter values are configurable and can be adjusted to suit the energy and accuracy requirements of mobile phone applications.

3.6 Related Work

Much research work has concentrated on inferring various kinds of social-oriented contextual information using smartphones [GLC+08, LYL+10, KLNA09, ROPT05] such as inferring speaker [MCR+10, LBBP+11] or capturing activities [MLF+08, RMB+10, WLA+09]. Systems that use purpose-built mobile devices to autonomously measure the user's contextual information include the Sociometer (or sociometric badge) [CP03] and the Mobile Sensing Platform [CBC+08].

Energy is a vital issue when building mobile sensing systems. The EEMSS platform [WLA+09] uses a hierarchical sensor management strategy to recognise participants' activities and achieves energy savings by using a minimal set of sensors and heuristically determining sampling lengths and intervals for the sensors to detect the user's state as well as transitions to new states. The duty cycling parameters should be refined offline through empirical tests and the system uses fixed fine-tuned intervals. Unlike the EEMSS system, the proposed adaptive sensing scheme dynamically adjusts the sensor duty cycling intervals according to the user's context. Further, we have shown in this chapter that the proposed dynamic scheme conserves more energy than static schemes.

The Acquisitional Context Engine (ACE) [Nat12] is a middleware for mobile sensing applications that supports continuous context recognition while achieving high energy savings. ACE automatically learns relations between the various context attributes, for example, if the user is at home then she is not driving, and exploits these to save energy. *SeeMon* [KLJ+08] is a context monitoring service for mobile devices based on several sensors and it achieves energy efficiency by sensing from a minimal set of sensors. These approaches achieve energy savings by inferring a context attribute using data from an inexpensive, or lesser number of sensors, or already available information, whereas the proposed scheme achieves energy savings by dynamically adjusting the sensor sampling interval. Duty cycling to achieve power savings is also a widely used technique in many mobile sensing systems [WLA+09, MLF+08, KLGT09]. Most of these systems, however, use static intervals which, as shown in this chapter, is inefficient.

The Jigsaw continuous sensing engine [LYL+10] balances the performance requirements of applications and the resource demands of continuously sensing on the mobile phone. It supports accelerometer, microphone, and GPS sensors. The system implements a dynamic duty cycling component that achieves energy savings by decaying the microphone sampling frequency (i.e., increasing the sampling interval) when no acoustic event is captured. However, the authors did not present exploration of the energy-accuracy trade-offs of the dynamic duty cycling method. In the case of the GPS sensor, they use a scheme based on the discrete-time Markov Decision Process (MDP) to learn the optimal GPS duty cycles. Their target is to learn the optimal duty cycling policy considering battery budget, duration of activity, and the mobility pattern of the user. The proposed adaptive sensing scheme uses a reinforcement learning mechanism where we consider the history of events in adjusting the probability of sensing, *i.e.*, more the number of consecutive interesting events higher will be the probability of sensing (as the sampling should be aggressive if a continuous stream of interesting events have been detected), whereas in a Markov Model the decision taken at a state is generally independent of the previous states. The authors of [CGS+09] also considered the energy budget and in particular, given an energy budget they try to minimise the *average localisation error* using a dynamic programming approach. However, the formulation based on energy budget differs from the proposed methodology in this chapter, i.e., dynamic adjustment of the sleep interval between consecutive sensing windows according to the user's context.

Nericell [MPR08] is a mobile sensing platform that monitors road and traffic conditions, and it achieves energy efficiency by employing the concept of *triggered sensing*, where an inexpensive sensor is used to trigger the operation of an expensive sensor. Triggered sensing has also been used in the SenseLess system [BAPH09]. Although triggered sensing can be configured using the rules framework presented in this chapter (Section 3.3), the adaptive scheme does not depend on the triggered sensing. The proposed approach relies on dynamically learning the user's behaviour and adjusting the sensor duty cycling intervals accordingly.

Virtual Compass [BAB+10] builds a relative neighbour graph using radio technologies like Bluetooth and Wi-Fi, and can be a useful tool for social applications. It achieves energy efficiency by adapting scanning rates and monitoring topology changes, and selecting the most appropriate radio interface according to the energy characteristics. In [RPS+10] the authors address the problem of energy-delay trade-offs in smart phone applications and in particular, they focus on delay-tolerant applications. They present *SALSA*, an algorithm based on Lyapunov optimisation, which achieves energy efficiency by adapting to channel conditions and deferring data transmissions accordingly. Llama [BRC+07] is an energy management system based on user statistics in terms of usage and recharge cycles, and exploits excessive energy for a better user experience. Finally, several energy-saving schemes for mobile devices were discussed and compared in [VBH03]. Although these schemes address the problem of energy consumption on mobile platforms, they do not address the energy-accuracy trade-offs of sensor sampling in mobile phones.

3.7 Conclusions

In this chapter we discussed a design method to adapt sensor sampling intervals to the context of the user. The design includes identifying interesting and uninteresting events and adjusting the sampling interval accordingly based on advance and back-off functions. We also presented a learning based approach that can be used in these functions. The learning scheme dynamically adapts the sensor sampling interval to the context of the user using a probabilistic approach. We also presented a rules framework to support the writing of rules to trigger the sensor sampling based on the user's context. We then designed APIs for the adaptive sensing scheme so that mobile phone applications that aim at energy savings can utilise the services of the adaptive framework.

The results showed that the learning based scheme saves considerably more energy compared to the continuous sensing scheme, while achieving almost the same level of accuracy. In particular, we showed that the proposed adaptive scheme uses 50%, 42%, and 43% less energy compared with the continuous sensing scheme for the Bluetooth, accelerometer, and microphone sensors, respectively. We further showed that it achieves much higher accuracy than a fixed 50% duty cycling scheme for comparable (microphone) or better energy savings (Bluetooth, accelerometer). We also explored the energy and accuracy results of varying the α_I and α_D values that can be adjusted according to the requirements of mobile phone applications. In the next chapter we will show that the adaptive sensing scheme can be combined with other schemes to further increase energy savings.

4 Sensing Offloading

4.1 Introduction

In the previous chapter we discussed an adaptive sensor sampling scheme and showed that it achieves significant energy savings. We have, however, also showed that energy needs to be expended in querying the phone's sensors often to achieve high accuracy. Although the adaptive sampling technique improves energy savings, there is still need for more efficient solutions before energy-hungry sensing applications can be widely accepted by everyday users. In this chapter we introduce a new approach that offers significantly bigger reductions in the phone's energy consumption without compromising the accuracy of event detection by *opportunistically offloading sensing to fixed sensors embedded in the environment if available.* Offloading sensing to infrastructure sensors is another step towards answering *Research Question 1 (How can we accurately capture raw data from the sensors in smartphones in an energy-efficient way?)* described in Chapter 1.

Most modern buildings are instrumented with a variety of sensors such as RFID access control systems, light sensors, motion sensors etc. Mobile phone applications can take advantage of the presence of such sensing infrastructure to reduce the energy cost of detecting social activities. By allowing mobile phone applications to interact with sensors in the environment whenever they are available, there is an opportunity to design rich systems that can maintain highly accurate sensing using both local phone and infrastructure sensors without compromising the battery life. Although offloading reduces the cost of sensing on the phone, it requires additional energy in the form of network communication between the phone and the sensor. The total energy cost incurred on the phone depends on how frequently the phone is expected to communicate with the sensor, which depends both on the user's mobility pattern, and on the rate at which events are detected by environmental sensors. A key challenge in supporting sensing offloading is to design a smart offloading scheme for optimal energy performance without compromising sensing accuracy. We note that offloading sensing to remote sensors may save energy on the phone, however, it incurs additional energy in the sensing infrastructure. Our aim is to save energy on the mobile phone as it is powered by a limited battery source and we do not consider the energy consumed by the sensing infrastructure as they are generally powered by their own energy supply stations with always-on power supply.

In order to offload sensing to the remote sensors, smartphones should be able to discover the services and capabilities of the sensing infrastructure. Several works [ZMN05, Dar10] have addressed the issues related to the service and device discovery. Jini network technology, Microsoft's Universal Plug and Play (UPnP), and Service Location Protocol (SLP) [Ric00] are some of the technologies that can be used to support service discovery and advertisements. In this chapter, our main focus is on designing an efficient sensing offloading scheme and we assume the presence of a discovery service to find the sensors in the environment. In order to communicate with the infrastructure sensors, we adopt an existing protocol, which will be detailed further in the chapter.

In this chapter we present a novel sensing offloading scheme that smartly switches between the local phone sensors and the sensors in the environment, when available, to achieve energy efficiency while delivering accurate data to the applications. The sensing offloading scheme takes into account parameters like sensor type, sampling rate, event rate, network connection state, and the mobility patterns of users to make a decision about *whether to use local phone or remote sensors*. We also present the design and implementation of a sensing platform that uses the sensing offloading scheme to offer seamless transition between local and remote sensing for mobile phones, in order to support accurate sensing of social activities, and offers a simple social sensing API to mobile phone applications. The system targets primarily office environments, where certain sensing modalities (i.e., room occupancy sensing) are commonly available.

Chapter outline. In Section 4.3 we present the architecture of a system that utilises the sensing offloading scheme and handles interactions between the mobile device and sensors in the environment. Sections 4.4 through 4.6 describe our experimental approach to devise an efficient *Sensing Offloading Scheme* scheme that achieves high-energy efficiency by combining both local and remote sensing. In Section 4.7 we evaluate our sensing distribution scheme using a number of benchmark tests. 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 and only remote sensors are used, respectively. Furthermore, we show through real phone-based tests (in Chapter 6) that the scheme extends battery life by more than 35% compared to when no sensing offloading to the infrastructure is used. We present related work in Section 4.8 and finally, we present conclusions in Section 4.9.

4.2 Motivation

The energy performance of mobile phone sensing applications can be significantly improved if costly sensing tasks are offloaded to sensors that are embedded in the environment. Users of a mobile sensing application can often find themselves in close proximity to sensors that can help alleviate the burden of continuous sensing. For example, a door sensor in an office can be tasked to send a notification of door activity (that indicates a change in the number of people present in the room), and allow a mobile application to suspend continuous scanning for co-located people. Another example is, by relying on a building's access control system, a mobile phone application can decide to suspend any location tracking mechanisms on the phone while the user remains in the same building or even room. In situations where appropriate sensing in the environment is not available, the phone can fall back on traditional mobile phone sensing techniques. Considering the typical living patterns of most users in which a vast proportion of their daily lives is spent at home or in a working environment, the sensing offloading approach could achieve significant energy gains while enabling the operation of accurate social sensing applications on mobile devices.

Offloading opportunities may not be available at all times. Indeed, in a typical office environment sensing infrastructure can be sparse. The possible gain in mobile phone battery life is an incentive to the instrumentation of an environment with an appropriate sensing framework. Further, in office environments, the usability and usefulness of specific applications such as interaction and collaboration monitoring can be improved by capturing data from more diverse sensor streams (e.g., door/desk sensors) that are generally not available on smartphones. However, we do not make any assumptions that sensing infrastructure is available in all the locations at all times. Although fully instrumenting an environment with appropriate sensors can increase the battery life of mobile phones, such drastic and costly solutions may not always be practical. Our approach considers mobile applications that adapt seamlessly to the availability of sensing infrastructure in the environment, offloading sensing when possible, and relying on local sensing when no such infrastructure is available. This novel approach has the potential to lead to highly accurate mobile sensing with minimal impact on the mobile phone's battery life. We envisage a model where sensor providers in buildings can allow the sharing of their sensors with mobile phone applications through well-defined standards. The incentives for this are in billing through the mobile phone application using the sensors. Clearly, a privacy framework for sharing and communication should be put in place, but many existing solutions can be adopted to support this [LEMC12].

Designing a scheme for opportunistically offloading sensing to the environment poses a number of challenges.

- Firstly mobile applications are typically designed without any prior knowledge of or access to the target environment in which they will operate. Any feasible solution should therefore assume environments that are not designed to serve this particular functionality, relying on widely adopted communication protocols for interacting with existing sensing infrastructure.
- Secondly, even if appropriate sensing modalities are available in a particular environment, offloading sensing may not always be the most efficient solution. Offloading can at times cost more than local sensing. Although remote sensing can reduce the cost of actual sensing on the device, it imposes an increased cost in the form of network traffic. In general the overall cost is a function of a number of variables: sensor type, sampling rate, event rate, and network energy.
- Thirdly, the user's mobility patterns play a crucial role in offloading sensing tasks.

We address these challenges in the remainder of this chapter.

4.3 System Architecture to Support Offloading

In this section we present the architecture of a system that can perform remote sensing to capture sensor data. The system supports the sensing modalities presented in the previous chapter i.e., accelerometer, Bluetooth, and microphone. Specifically, we design a mobile phone service supporting social sensing for mobile applications. When sensing from the local phone sensors, the system may utilise the services of the adaptive sensing component described in the previous chapter. A key feature of the system is its transparent support for opportunistic offloading of sensing to sensors embedded in the environment, with the aim of improving energy efficiency without hindering accuracy. The design of the sensing offloading scheme used in the system will be further described in this chapter (Section 4.6). The operation of the system 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, the system falls back to local sensing utilising the resources of the mobile device. The supported sensing modalities are exposed to the mobile applications through the *Social Sensing API*. The API allows applications to receive sensing notifications in an asynchronous manner, similarly to the reporting of location updates provided by the Android Location Services. The architecture of the system is shown in Figure 4.1.

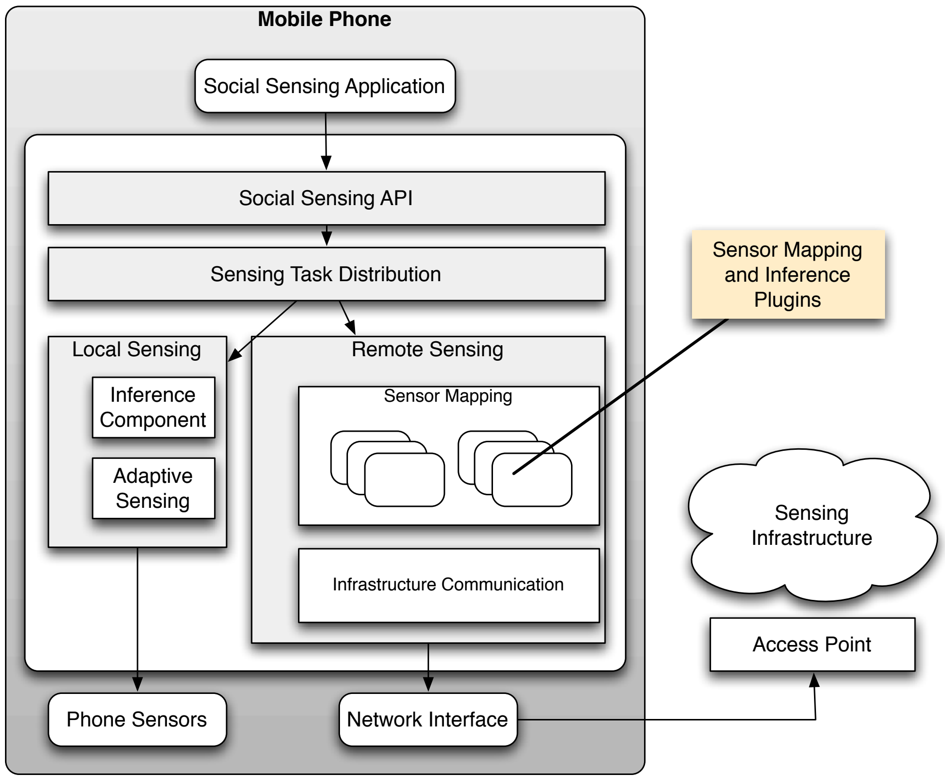


Figure 4.1: Architecture of a system utilising adaptive sensing, and sensing offloading.

4.3.1 Interacting with Sensing Infrastructure

One of the key enabling technologies for the sensing offloading scheme is the emergence of a range of Web-based architectures that allow interactions with sensing infrastructure over IP networks. We decided to assume the presence of sensing infrastructure that follow the same architectural principles that are adopted by such architectures. Systems such as SenseWeb [KNLZ07] and Sensei [VBH+10] identify two key elements in their architecture: the presence of a rendezvous point that 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 our system 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 the system we adopt the sMAP [DHJT+10] like communication protocol for interaction with specific sensing resources. sMAP defines a REST-full/JSON based communication protocol. Publish-subscribe communication is available through the `/reporting` interface. Subscription to specific sensing events can be achieved by sending a POST request to that interface:

```
POST: http://dom.ain.org/report
{
  "ReportResource" :
    "http://dom.ain.org/data/sensing/ACC_D1_R01/*/reading",
  "ReportDeliveryLocation" : "http://10.10.0.1:5001/",
  "Period" : 60, "Minimum" : 50, "Maximum" : 100
}
```

The information that can be retrieved through the rendezvous point plays a key role in the operation of our system. The expectation is that the system can identify the physical location of sensor points. In the design of the system 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 (Figure 4.2). 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 [BR07]. The system can be trivially extended with additional schema parsers to support multiple infrastructure interfaces.

Such information is crucial in order for the system to identify what types of sensors are available around the user at any given time. Furthermore, an important requirement for the accurate operation of the system is that a given environment should offer an accurate indoor localisation technology 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, the system will attempt to retrieve the sensing infrastructure model from a well-known repository. If the building offers a known indoor localisation technology (i.e., Wi-Fi fingerprinting, Bluetooth RSSI trilateration), the system will utilise this technology to locate the user in the building. That information is then used to discover the types of sensor that can be accessed by the platform at any given time.

Sensor Mapping

The information that the system collects from the infrastructure is specific to the sensing device accessed. The system allows the interpretation of such data 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 range 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

```
<?xml version="1.0" encoding="UTF-8"?>
<mt:METIS xmlns:mt="http://dom.ain.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:SensorList>
 </mt:SensorNetwork>
</mt:METIS>
```

Figure 4.2: Sample sensing infrastructure manifest obtained by the system from a service provider.

design of such plug-ins is non-intuitive and depends on the presence of specific sensors in the environment. As our main aim 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 significant 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 on the location of the user, or the current activity state, for example.

Plug-ins that have been implemented for the system include:

- If real-time room occupancy information is available, subscribe to receive events about the current room, and switch off 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.

4.4 Approach

The performance of the system architecture presented in the previous section relies mainly on the design of a sensing task distribution strategy that is able to select the most energy efficient method of sensing (local phone or remote sensing) in any given situation. Simply offloading sensing tasks to the infrastructure may not always be the most efficient solution: although offloading reduces the cost of sensing on the mobile device, at the same time it imposes additional energy cost in the form of network communication. The actual communication cost can vary significantly depending on the user's circumstances and the environment's: the mobility patterns of the user, the rate at which events are detected by environmental sensors, and sensing parameters like the cost of sampling a sensor and sampling rate. An efficient offloading strategy should be able to dynamically adapt in the face of changing circumstances, by evaluating the expected cost-benefit trade-off in deciding to perform sensing offloading.

In order to design the *Sensing Offloading* scheme we followed an experimental approach. We performed a test deployment in which we collected information about user behaviour and available sensor readings within an office environment. We then designed a generic sensing offloading scheme and evaluated it through several benchmark tests using the collected traces. Finally, through the implementation and real deployment of a social sensing application for the workplace, we evaluated the sensing task distribution scheme's performance in action. The overall methodology is summarised as follows:

- **Data Collection Deployment.** The aim of this deployment was to collect sensor readings with continuous sampling by both the mobile phones and already deployed fixed sensors in the office environment. We used high sampling rates for all sensor data. These two datasets allowed us to capture some ground truth on sensor data both from mobile phones and from sensors embedded in the environment.
- **Sensing Offloading Design.** Using the traces from the initial deployment, we experimented with a range of sensing distribution policies. The aim of this analysis was to identify the parameters that affect energy cost and accuracy when deciding to offload sensing from a mobile phone to infrastructure sensors.

- **System Deployment.** By utilising the services of the proposed platform (i.e., sensing offloading), mobile applications will be able to perform social sensing efficiently. We developed a social application and deployed it within our research institution with 11 participants for over a month. We will present the results on the application aspects of the deployment in Chapter 6, which also covers other social psychological applications that we built. We present the results on energy in this chapter.

4.5 Data Collection Deployment

We performed the initial deployment in the office space of our institution. The aim was to collect data traces both from mobile phones carried by users, and from sensors embedded in the environment. Although continuous sampling may be redundant and inefficient, it is the best way to acquire the most comprehensive dataset capturing a phenomenon. Our deployment accordingly was one where all sensors, including the users' phones, in our office environment were used and sampled at very high rates. The data collected through this deployment was intended to be used to understand the opportunities for offloading sensing from the phones to the environment and to evaluate the performance of this technique. In the next subsections, we present details of the sensing modalities used in the phones and infrastructure. We note that we could not use the dataset that has been described in the previous chapter (Chapter 3, Section 3.5.1) for evaluating the sensing offloading scheme as it has traces only from the sensors of the participants' smartphones and not from infrastructure sensors. We, therefore, performed a different data collection deployment that will be described in this section for exploring the sensing offloading.

4.5.1 Mobile Phone Sensing

We designed an Android application to perform indoor localisation, co-location detection, and conversation detection using the mobile phone's microphone. The classifiers implemented are the same as those described in the previous chapter (Section 3.5). The application was implemented in Java on the Android 2.3.3 platform.

The sensor sampling was implemented to gather data from the Bluetooth and microphone sensors. The Android Bluetooth APIs were used to discover Bluetooth devices in proximity. The discovery process involves an inquiry scan followed by a page scan of each found device to retrieve its Bluetooth name, Bluetooth MAC address, and Received Signal Strength Indication (RSSI) value. The conversation recognition module is based on that used in the previous chapter on adaptive sensing, which was implemented using the Hidden Markov Model Toolkit (HTK) [HTK] (Section 3.5). More details about this classifier will be presented in Chapter 6 (Section 6.2).

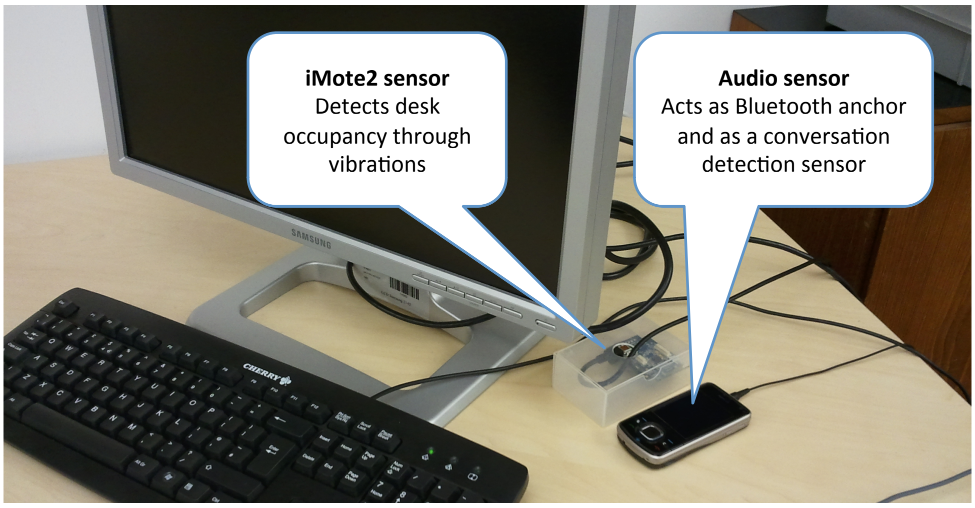

iMote2 sensor
Detects desk occupancy through vibrations
Audio sensor
Acts as Bluetooth anchor and as a conversation detection sensor

Figure 4.3: Sensing infrastructure: Each desk is instrumented with a desk occupancy sensor and a Nokia 6210 Navigator phone acting as conversation detection sensor and bluetooth anchor point.

4.5.2 Sensing Infrastructure

In our deployment we aimed to exploit existing sensing infrastructure in our environment. As part of previous sensing experiments within our research institution [ELP+12] two sensing infrastructures were already available: an indoor localisation infrastructure and a room occupancy sensing application (Figure 4.3). The indoor localisation system relied on Bluetooth anchor points deployed around the building. In particular Nokia 6210 Navigator phones were set to act as Bluetooth anchors and to assist in the localisation of mobile phones. The infrastructure included 12 such Bluetooth anchor points covering a space of 10 offices. Furthermore, each Nokia node performed a periodic Bluetooth scan using the *lightblue* module for *Python for S60* (PyS60).

As part of an infrastructure to capture accurate room occupancy data, a network of imote2 sensors had already been deployed around the office spaces. The sensors were attached to desks and were able to detect when a particular desk was occupied. The desk occupancy status was inferred by detecting vibration patterns using the 3-axis accelerometer sensor embedded in the node. Each of the nodes periodically sent its current state (i.e., whether the desk was occupied or not) to a root node connected to a server. Desk usage events were sent to the root node using the Collection Tree Protocol (CTP) [GFJ+09]. The overall network consisted of 13 nodes covering a space of 10 offices. In order to offer additional support for conversation detection we enhanced the capabilities of the Nokia 6210 Navigator devices acting as Bluetooth anchor points with conversation sensing functionality. We implemented the same detection algorithm that was running on the user's mobile devices (i.e. the Hidden Markov Model Toolkit based conversation detection) and exposed the captured information as sensing events reporting conversations or silence that were sent to a back-end server.

4.5.3 Lessons Learned

The initial deployment involved 10 users and was in operation for one working week. During the deployment each user carried a smartphone while additional data were collected from the static sensors installed in their working environment. The first result of this deployment was an estimation of the effect of continuous sensing on the phone's battery consumption. We measured the battery drain of the phone on two Samsung Galaxy S phones, while the phone was not used for any other purpose. The battery drain was measured using the *BatteryManager* API [1] of the Android platform, and Figure 4.4 shows the result. We can observe that the phone lasted around 24 hours with only the data collection application running. Considering that the battery life would also be hampered by the normal phone usage by the participant, the typical battery life experienced by the users would have been significantly less.

By analysing the users' mobility patterns we tried to estimate the opportunities for sensing offloading. According to the traces, users spent 64% of their time inside their own office and 21% of their time in other offices (Figure 4.5) with sensing capabilities. These were promising results showing that on average, users would be in an environment where sensing offloading could be used 85% of the time. 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 Figure 4.6 we observe that users visited other rooms for short (6-minute) intervals (e.g., to have a chat with someone or to have a short discussion about work). To understand the frequency and duration of such events we plotted the CDF of how much time each user spent when entering a room (Figure 4.7): there is a high number of short visits (50% of them last less than 20 minutes). Such short visits could cause either lower accuracy or inefficiency when offloading sensing to the environment. A short visit to a room where the mobile phone immediately offloads sensing, while the user leaves shortly after, could cause sensing events to be either missed or mistaken (reported by the wrong environment). Finally, users spent 15% of their time in locations that were not instrumented (Figure 4.5), further justifying our approach, which does not only rely on fixed building instrumentation, but exploits the portability of mobile phone sensing.

4.6 Sensing Offloading

Having analysed the results of the test deployment, we are now 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 to perform offloading is based on the estimation of the energy cost when sensing is performed locally on the phone, and the prediction of the cost when offloading is performed remotely in the cloud. In estimating

[1]http://developer.android.com/reference/android/os/BatteryManager.html

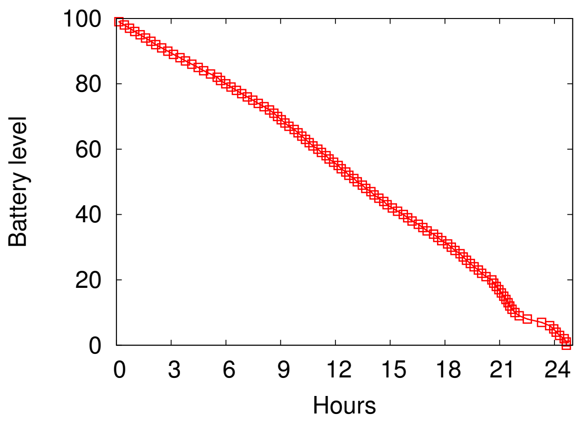


Figure 4.4: Battery drain using phone sensing.

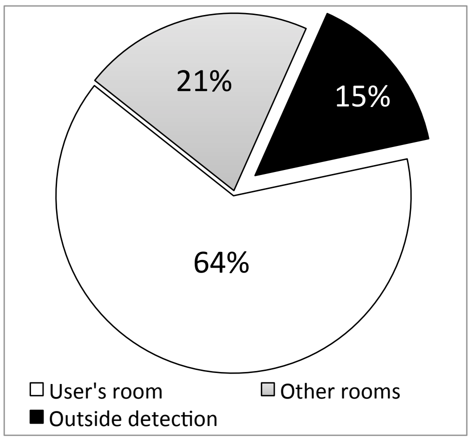


Figure 4.5: Percentage of time spent by users.

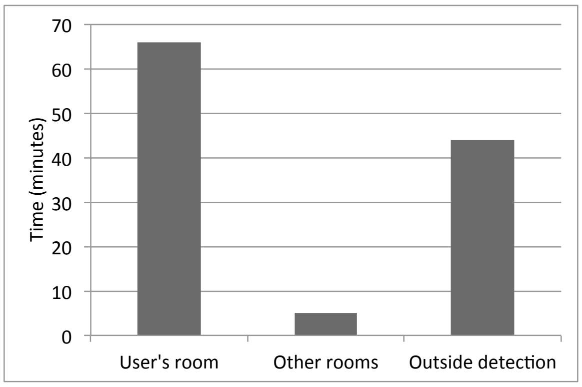


Figure 4.6: Average time spent for each visit to a location.

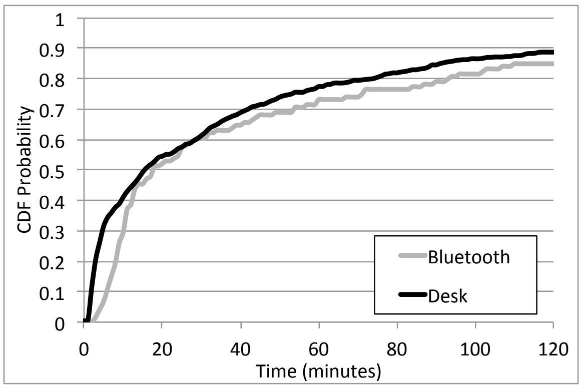


Figure 4.7: CDF of time spent for each location visit.

the latter, the mobility pattern of the user, the time they spend in a particular location, and the rate at which events are reported by the remote sensor are factors that affect the energy cost. In the following sections we describe an offloading scheme that achieves energy efficiency by considering all these parameters.

Offloading a sensing task from the user's mobile phone to the infrastructure involves delegating them to one or more sensors in the environment: the infrastructure will then be responsible for monitoring for matching events and the phone will just wait for possible notifications. For instance, in the case of detecting a conversation, the phone's microphone would be used to capture audio and test whether a conversation is detected. Offloading such an activity involves a subscription for events by a suitable sensor in the environment and then listening for network notifications when conversations are detected by the infrastructure. Such an offloading scheme may save significant energy as the phone need not constantly sample the sensors and process the data for possible events. However, offloading introduces network overhead (Wi-Fi connection maintenance and *control traffic*) needed to manage offloaded tasks and to receive notifications about possible events.

Therefore, if local sensing is cheap or users are highly mobile then this control traffic might be uneconomical. Furthermore, in high-mobility scenarios offloading a task may result in lower accuracy as the user might have already moved outside the infrastructure's sensing range by the time the new location is detected and the task would be transferred back to the phone.

The decision to offload a sensing task only considers sensors that are near the user's current location. The definition of "near" is context-dependent and it is the distance from the sensor in the infrastructure where sensing can be offloaded to the environment without considerably sacrificing the accuracy of inferences. Most environments are divided into designated areas, office spaces, or rooms. These may have physical boundaries such as closed rooms with doors, or they can be logically divided, for example, cubicle spaces. The assumption is that these spaces can be small enough to allow the offloading of sensing by users located in that space without any impact on the accuracy of the detected information. When a user moves to a different location, offloading needs to be re-evaluated. An ideal offloading scheme reduces control traffic and energy consumption, while offloading tasks to sensors that are in close proximity to (in the same environment as) the user.

The mobility patterns of users across different environments vary significantly based on many factors like type of work performed and time of the day. Clearly a dynamic scheme is required that takes into account historical information about mobility to decide on offloading a given sensing task to reduce the overall energy consumption without compromising accuracy. We design *gain threshold based offloading* scheme that uses historical data about the mobility patterns and information about the estimated energy cost of the sensing task and the control traffic. The energy costs could be measured on the Android phones using a power meter [ZTQ+10] (such as the Monsoon Power Monitor[2]) and on the Nokia phones using the Nokia Energy Profiler. We compare this approach with the two extreme scenarios of *always offloading* and *never offloading* (local phone-based sensing).

4.6.1 Gain Threshold based Offloading

The *Gain Threshold* scheme operates by calculating the probability that offloading a particular sensing task would result in a gain in terms of energy consumption 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 (less than 50% chance of resulting in loss) then the offloading is performed.

The probability of gain used for decision making is calculated by estimating the possible communication costs that the phone may incur if offloading is performed. When a sensing task is offloaded to a remote sensor there are two types of energy costs involved, *fixed* and *variable* costs.

[2]http://www.msoon.com/LabEquipment/PowerMonitor

- The fixed costs are the costs involved in maintaining a network connection between the phone and the infrastructure provider system, subscribing to a remote service for offloading, and cancelling 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's behaviour (mobility, interaction etc.).

A decision to offload a sensing task results in energy gain if the user stays in the location for long enough that the sum of 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 (e.g. this could be measured on the Nokia phones using 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 the specification of the gain threshold we use the following notation:

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

According to the definition of *gainTimeThreshold*, for a sensor s, the total 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 positive energy gain, and if remote sensing is used for less than *gainTimeThreshold* amount of time, then it results in 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}) $\times$ Local sampling rate (S_{sl}) $\times$ 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}) $\times$ Cost of network transfer per update (C_{sr}) $\times$ G_s) + Fixed 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., $L_s = R_s$.

$$\implies C_{sl} \times S_{sl} \times G_s = C_{so} + C_{sr} \times U_{sr} \times G_s + \frac{C_{nr}}{N} \times G_s \tag{4.1}$$

$$\implies G_s = \frac{C_{so}}{C_{sl} \times S_{sl} - C_{sr} \times U_{sr} - \dfrac{C_{nr}}{N}} \tag{4.2}$$

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. In other words, it is the minimum amount of time the user should stay in the current location after the system offloads the sensing task, in order to achieve energy gain.

Probability of Gain Estimation

In this section 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 $\{v_{j_1}, v_{j_2}, \dots v_{j_k}\}$ be the total visits of the user to a location j, and let $\{tv_{j_1}, tv_{j_2} \dots tv_{j_k}\}$ be the total duration of each of the k visits, respectively. First, for each sensor s, we divide the total duration of the l^{th} visit to a room j into two parts: *favourable time* (ft_{sj_l}) and *unfavourable time* (ut_{sj_l}). Favourable time for a sensor is the time during which offloading of the sensing task results in a positive gain, and unfavourable time for a sensor is the time during which offloading results in a negative gain. Therefore, for a visit to a room, the favourable 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 unfavourable time is G_s, *i.e.*, for the l^{th} visit to a room j by the user, favourable and unfavourable times for a sensor s are calculated as:

$$ft_{sj_l} = tv_{j_l} - G_s \tag{4.3}$$

$$ut_{sj_l} = G_s \tag{4.4}$$

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 favourable time at location j for all visits divided by the total time at the location j.

$$\implies pt_{sj} = \frac{\sum_{l=1}^{k} ft_{sj_l}}{\sum_{i=1}^{k}(ft_{sj_i} + ut_{sj_i})} \tag{4.5}$$

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 cancelled (unsubscribed from remote sensing) 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.

The intuition behind this model is that if there is greater than 50% chance that a user stays in a location for more than G_s amount of time for a visit, then there is more than 50% chance that there will be gain, and it is beneficial to offload the sensing task, otherwise it will most likely result in more energy cost than local phone sensing. We present a detailed evaluation of this scheme through micro-benchmarks in Section 4.7.

4.6.2 Extensions of Gain Threshold Offloading Scheme

Behavioural 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 behaviour 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 minutes, however, they might only visit the common room for shorter periods and less regularly at other times of the day. User behaviour 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 behaviour 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:

$$p_{sj} = pt_{sj,t,d,c} \tag{4.6}$$

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 favourable and unfavourable 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:

$$p_{sj} = pt_{sj,c} \tag{4.7}$$

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.

4.7 Micro-Benchmarks

As discussed in the previous section, the proposed sensing offloading scheme smartly switches between the local phone sensing and remote infrastructure sensors, therefore, it is expected to achieve higher energy savings than the schemes that use only phone sensors or only remote sensors. In order to evaluate its performance and quantify the energy savings we compare the offloading scheme with two cases reflecting two opposing extremes in sensor offloading:

- **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 behaviour of an application that runs purely on the user's mobile phone or in an environment where no infrastructure sensing is available, similar to the model presented in Chapter 3. Even though this appears to be an inefficient scheme compared to opportunistic task offloading, there are cases where this is efficient: when the cost of local phone sensing is lower than the network energy consumption of data exchange with remote sensors, or in highly mobile scenarios where local phone sensing can potentially save more energy by avoiding the cost of unnecessary control traffic induced by continuous offloading and cancelling of sensing tasks. This method is essentially identical to our data collection deployment (Section 4.5) where no offloading was used.

- **Always Offload.** In this scheme the mobile phone offloads sensing every time the user is in an environment where appropriate sensors are available. Since this scheme utilises remote sensors and suspends local sensing, it may achieve more energy savings than local phone sensing. However, Figure 4.7 shows that there are many location change events that last only for a short time. We will show that this induces more control traffic through sensing hand-overs or reduce accuracy due to the missed events during these hand-overs.

The evaluation is performed with a number of benchmark tests using the Android emulator and the collected traces (presented in Section 4.5).

4.7.1 Dataset

We used the dataset collected from the initial deployment for benchmarking the sensing task distribution schemes. The dataset contains a working week of 10 users in an office environment. The data include indoor location and conversation status detected through their mobile phones and sensors in the environment. As the dataset was collected with a continuous sampling process from the mobile phones and the infrastructure sensors, we consider the data gathered as the ground truth and assess the performance of the offloading schemes presented over it. We consider the following sensing tasks for the benchmarks: Bluetooth scanning for inferring indoor localisation and microphone recording for inferring conversation status.

4.7.2 Methodology

For the evaluation we developed a framework that runs on the Android emulator and utilises the dataset to simulate the real scenarios. The energy consumption is estimated by multiplying the total usage of each resource (such as Bluetooth scanning) with its corresponding energy consumption value. The energy costs include the cost of local sensing and the cost induced by network traffic.

Sensing Modalities

The benchmarks consider two sensing modalities: indoor localisation using Bluetooth scanning and conversation detection using microphones. In the former case, desk occupancy sensors are used for offloading Bluetooth scanning when users are at their desk. In the latter, microphone sensors in the environment are used to offload conversation detection from phones. Location scanning can also be offloaded if we assume the presence of door sensors in the environment [HGDW12]. Although no such data was available in our traces, we were able to generate door sensor traces using the mobility traces of the participants. Evaluation results consider both scenarios where door sensor data is and is not available. The schemes *local, always* and *threshold* refer to the three main schemes that were described. Moreover, the schemes *always_office* and *threshold_office* refer to the behaviour of the system when no door sensor data is available.

Performance Metrics

We evaluated the performance of the sensing distribution schemes using the following metrics:

- *Local sensing resources usage*: Length of time the local phone sensing resources are used.
- *Network usage*: This is the amount of data sent and received over the network.
- *Energy consumption*: This metric is measured as the sum of the total energy consumption for local sensing, processing, network connection maintenance, and data exchange.

We calculate the resource usage and multiply it for each resource its corresponding power consumption value to estimate the energy consumption. The power consumption values used in the evaluation are from the following works: [FKK11], [SP12], [RH10a], [RH10b] and [ZTQ+10]. The default sampling interval of local and remote sensors is set to 120 seconds. We pick the same sampling rate for both local and remote sensing paradigms in order to measure energy cost under similar accuracy conditions as reflected by the sampling rate.

Wi-Fi Baseline Cost

The benchmarks evaluated the behaviour of the schemes by considering the impact of enabling Wi-Fi on the mobile device if that was required by the operation of the proposed scheme. In estimating the true cost of remote sensing, it is reasonable to assume that the cost of maintaining a Wi-Fi connection (*Wi-Fi Baseline Cost*) is part of the cost of performing remote sensing. However, in a realistic setting, users typically enable Wi-Fi for other purposes. In order to analyse both situations, we benchmarked the behaviour of the schemes under both conditions: (i) measurements without considering the Wi-Fi baseline cost, assuming that Wi-Fi is already enabled by the user, (ii) measurements including the Wi-Fi baseline cost, where the Wi-Fi is enabled by the offloading scheme in order to access the sensing infrastructure. In both cases the network cost measurements include the energy overheads that are caused by the network traffic generated by the remote sensing scheme. In these measurements we approximate the network energy cost as a function of the network traffic that is generated, in accordance with the results reported in [RH10a]. The network traffic is estimated as an average of the number of bytes that are exchanged when performing a subscription or receiving an event (this includes the traffic for establishing a TCP connection, exchanging HTTP/JSON messages, and closing the connection).

Mobility

As discussed earlier in this chapter, the rate of remote sensor events depends on the user's mobility patterns. Therefore, we studied the effect of mobility on the energy cost of the schemes.

Sensing Parameters

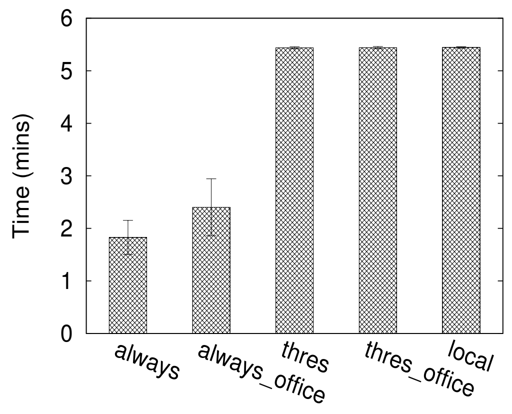
Since the sensing parameters such as sampling rate and the cost of sensing have a direct impact on the energy cost for local sensing, we studied the impact that these parameters have on the performance of the schemes. Finally, we explored the behaviour of the schemes when combined with the adaptive sensing technique that has been presented in the previous chapter (Chapter 3).

4.7.3 Results

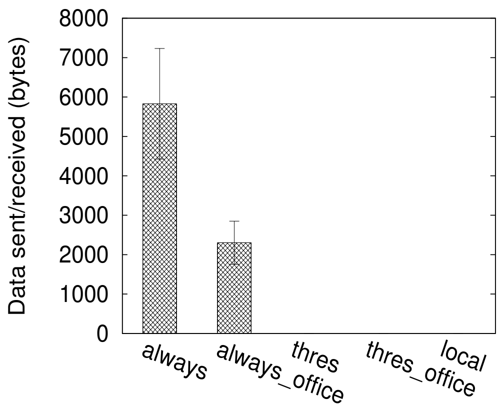
We first present the results which include the Wi-Fi Baseline Cost in the calculations. Then we describe the same results without these costs. In both cases the results are measured for the Bluetooth localisation sensing, and the combination of localisation and conversation sensing. In addition, we also evaluated the proposed scheme by varying the cost of sensing to show that the results are applicable to other sensing modalities.

Results Including Wi-Fi Baseline Costs

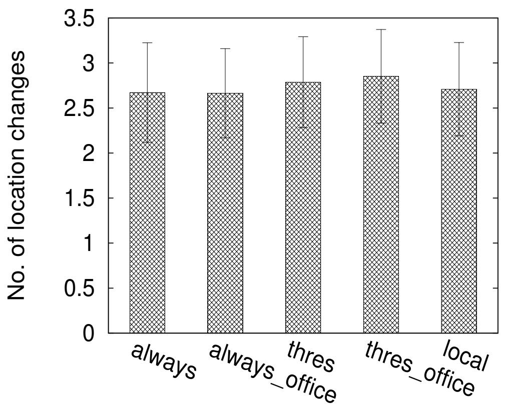
Firstly, we evaluated the performance of the schemes when Bluetooth localisation is the only sensing task active. Figure 4.8(a) shows the average Bluetooth scan time per hour for the offloading schemes. Since local sensing does not exploit sensors in the infrastructure, we can observe that it scans the local Bluetooth more often than the other schemes. The amount of scanning is less for the *always offload* scheme as it exploits the remote sensors, backing off when scanning is not necessary. The error bars in all the figures in this section represent the confidence interval computed considering the average value for each user as a data point. Figure 4.8(b) shows the average amount of data exchanged over the network per hour. The *Local* sensing scheme does not send any data to the server, and the *always offload* scheme uses a large amount of data traffic as it always tries to offload when there



(a) Bluetooth scan time.



(b) Network data sent/received.



(c) Number of location changes detected.

Figure 4.8: Local phone resource usage, and changes detected per hour for location detection.

are sensors in the environment. Since the *office* based schemes offload only when the user is in his/her office (based on desk sensors), the local sensing usage is more and network usage is less compared to the always offloading scheme. The *threshold* scheme behaves like local sensing and does not send any data to the server, indicating that the scheme never crosses the threshold to trigger sensing offloading.

In order to compare the energy consumption of the various schemes in achieving a similar level of accuracy, we fixed the accuracy by adjusting the sampling rate of both local and remote sensing. Figure 4.8(c) shows that the level of accuracy achieved in detecting the change of location is very close for all the techniques. We then compared the energy consumption of the schemes to achieve this accuracy.

Figure 4.9 shows the energy consumption for local Bluetooth sensing, sending/receiving data over the network, and total energy consumption per hour including the Wi-Fi baseline costs. The energy consumption for local sensing is higher for the local phone sensing scheme due to the significant amount of local sensing resource usage. The network energy

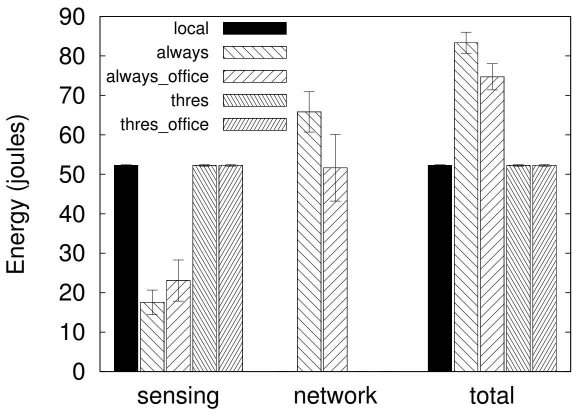


Figure 4.9: Energy consumption per hour for location detection considering Wi-Fi baseline cost.

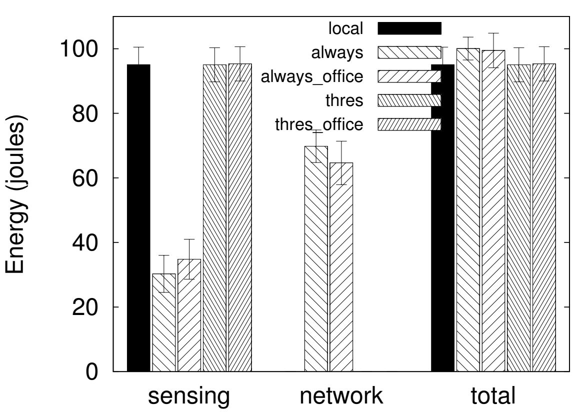


Figure 4.10: Energy per hour for location and conversation detection considering Wi-Fi baseline cost.

costs for the always offload scheme are higher compared to the local and threshold schemes. However, with respect to the total energy consumption, local sensing performs better. We can observe that the threshold scheme resembles the local phone sensing scheme, as in this case the Wi-Fi costs surpass the cost of local sensing. This result shows that the threshold scheme saves 37% energy compared to that of the always offload sensing scheme, and is the best performing scheme along with the local sensing scheme. We note that the always offload scheme occasionally uses local resources due to unavailability of sensing infrastructure in some locations.

We further enabled both Bluetooth and microphone sensors and performed a similar evaluation. The energy results are shown in Figure 4.10. We observe similar results for local sensing cost, and network energy cost, however, the total energy cost of all the schemes are very close. This is because the addition of the microphone sensor increased the cost of local sensing, while it resulted in an incremental increase in the network cost of the always offload scheme. However, the threshold and local sensing schemes are efficient in this case too and save 6% energy compared to the always offload scheme. As a general finding, as the number of sensors that can be offloaded increases, eventually the cost of remote sensing will be lower than the local sensing cost. Since the threshold scheme uses the gain calculation to decide on offloading, in this case, it is expected that the scheme will switch from local sensing to remote sensing, which will be demonstrated further in this section.

Results Without Including Wi-Fi Baseline Costs

Here we consider the case where Wi-Fi is turned on by default (i.e. by the user), and therefore the energy cost of maintaining the Wi-Fi connection is not caused by the oper-

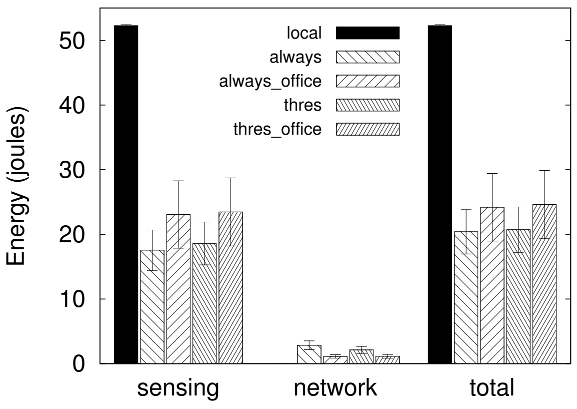


Figure 4.11: Energy consumption per hour for location detection without considering Wi-Fi baseline cost.

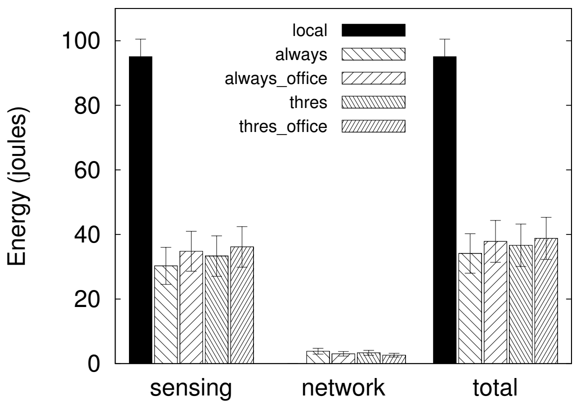


Figure 4.12: Energy per hour for location and conversation detection without considering Wi-Fi baseline cost.

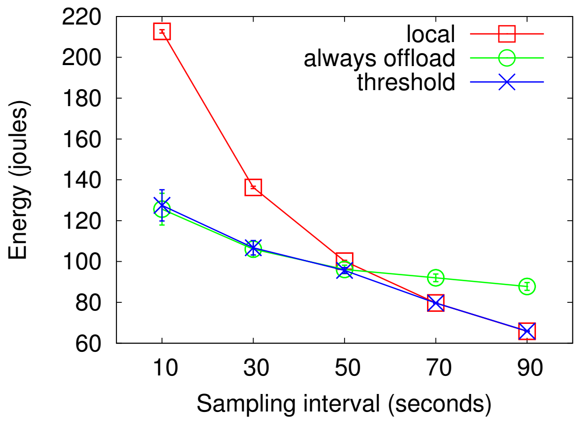
ation of the offloading scheme. In these benchmarks the network cost is considered only as a function of the additional traffic generated by the system.

Figure 4.11 shows the local sensing, network exchange, and total energy consumption of using the Bluetooth sensor for location detection (the microphone sensor is disabled). We now observe that the offloading schemes result in considerable energy savings as the Wi-Fi is already enabled and the sensing tasks are offloaded more aggressively. With respect to the overall energy consumption we can observe that the always offload scheme outperforms the local sensing scheme in this case as it exploits the sensing infrastructure. The threshold scheme resembles the remote sensing scheme, saving around 60% of energy when compared to the fully local sensing scheme.

We then enabled both Bluetooth and microphone sensors, and performed similar measurements. The results are shown in Figure 4.12. The local sensing cost and network energy cost results are similar to those of using only the Bluetooth sensor. With respect to the overall energy consumption, the offloading scheme performs much better than the local sensing scheme. In this case too, the threshold scheme (along with the always offload scheme) is the best-performing scheme and saves around 60% energy compared to fully local sensing.

Sampling Interval

As shown in the results so far, the threshold scheme tends to follow the optimal scheme in different circumstances. However, the optimal strategy does not depend solely on the state of the Wi-Fi connection. There are several other parameters that play a pivotal role, such as the sampling interval, the cost of local sensing, and the user's mobility patterns. The sampling interval is the interval between two consecutive sensor samplings and the



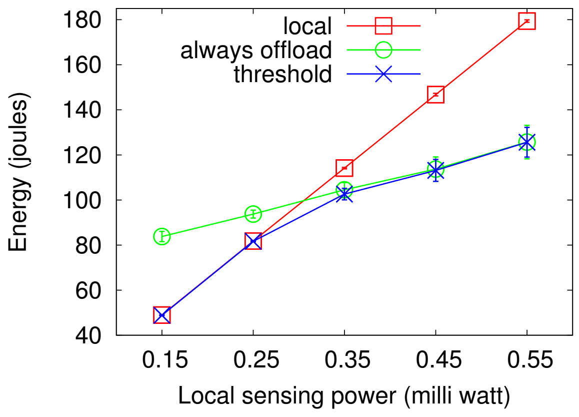


Figure 4.13: Total energy consumption per hour for location detection with varying sampling interval.

Figure 4.14: Total energy consumption per hour for location detection with varying sensing cost.

cost of local sensing is the power consumption to sample data from the local phone sensor. We evaluated the performance of the schemes with respect to sampling interval, sensing cost, and mobility variation. In this evaluation, we considered the Wi-Fi connection cost, i.e., Wi-Fi is off by default and the schemes should switch on Wi-Fi to use it. We restrict this evaluation to the always offload, fully local, and threshold schemes only as the other techniques based on *office-only offload* resemble their corresponding always offload schemes very closely.

Figure 4.13 shows the total energy consumption of the schemes with respect to sampling interval variation. We can observe that for low sampling intervals (high sampling rate), the optimal scheme is the always offload, while for high sampling intervals (low sampling rate) the optimal scheme is local sensing. This is due to the decrease in local sensing cost as a function of sampling frequency. However, the threshold scheme tends to match the best performing scheme in all cases, irrespective of the sampling interval.

Sensing Cost Variation

Another parameter that we considered in the evaluation of the offloading schemes was the impact of sensing cost on the performance of the schemes. Figure 4.14 shows the total energy consumption of the schemes with respect to local sensing cost variation. We can observe that for low power consumption values for local sensing, the fully local scheme is better and for higher power consumption values the always offload scheme is better. However, in this case too, the threshold scheme tends to follow the best performing scheme, irrespective of the local sensing power consumption.

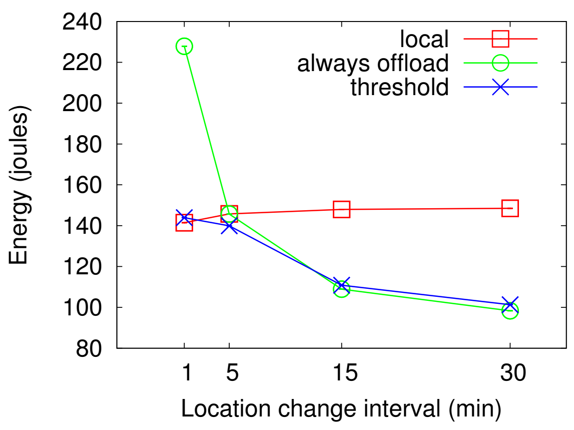


Figure 4.15: Total energy consumption per hour for location detection for varying mobility patterns.

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 sensors in the infrastructure. Both these costs depend on the behaviour of the users in the instrumented spaces. 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. As illustrated in Figure 4.15, the results demonstrate how high mobility can significantly increase the cost of sensing offloading if the user's mobility is not considered in 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. Since the proposed scheme considers the user's mobility patterns, it follows the best performing scheme.

As shown in the results so far, the threshold scheme tends to follow the optimal 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 that consider: 1) adaptive sensor sampling, and 2) multiple sensors with different power and data requirements.

Adaptive Sampling

The use of adaptive sampling is efficient in mobile sensing systems as demonstrated in the previous chapter. In adaptive sensing, the sampling rate changes in the face of changing behaviour of the user, i.e., the sampling rate is reduced when no observed events take place, and increased when observed events occur. We attempted to evaluate the impact on

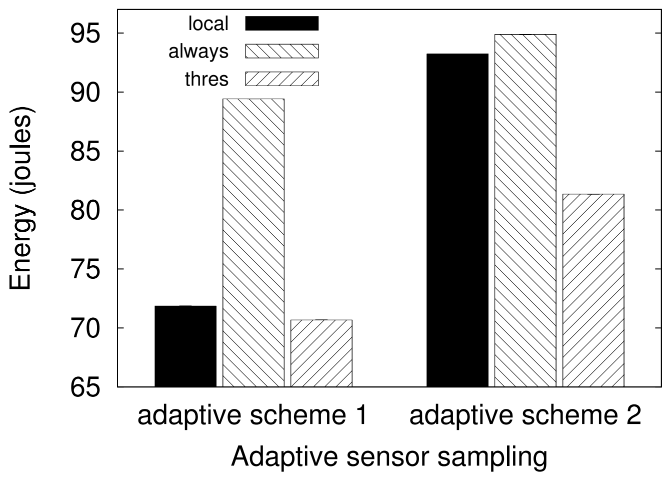


Figure 4.16: Energy consumption per hour with adaptive sampling schemes.

the energy consumption of the offloading schemes by using two simple adaptive sampling schemes presented in the previous chapter:

- In *adaptive scheme 1*, we used a linear advance and an exponential back-off of sampling interval i.e., *linear-exponential.*
- In *adaptive scheme 2*, we used an exponential advance and a linear back-off of sampling interval i.e., *exponential-linear.*

As discussed in the previous chapter, the sampling interval is increased using the back-off function when there is no change to the user's context and the advance function is used 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. In this scenario, when any of the schemes uses the local resources, the corresponding adaptive technique for local sensing is employed. Since our aim was to understand the performance of the schemes with respect to two adaptive sensing techniques, we chose static-adaptive schemes in both cases so that they have a comparable influence in both the tests. If we have chosen the learning scheme, then it would be difficult to differentiate whether the benefit in the case using the learning scheme was due to the dynamic adaptive sensing or the sensing offloading scheme.

The result of this evaluation is shown in Figure 4.16. We observe that for purely local sensing the exponential back-off (adaptive scheme 1) shows more energy savings when compared to the linear back-off (adaptive scheme 2). Also, the choice of adaptive scheme does not have a significant impact on the *always offload* scheme as this uses remote sensing most of the time (except in un-instrumented areas). However, in both cases neither the *always local* nor the *always offload* 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 optimum approach in every case.

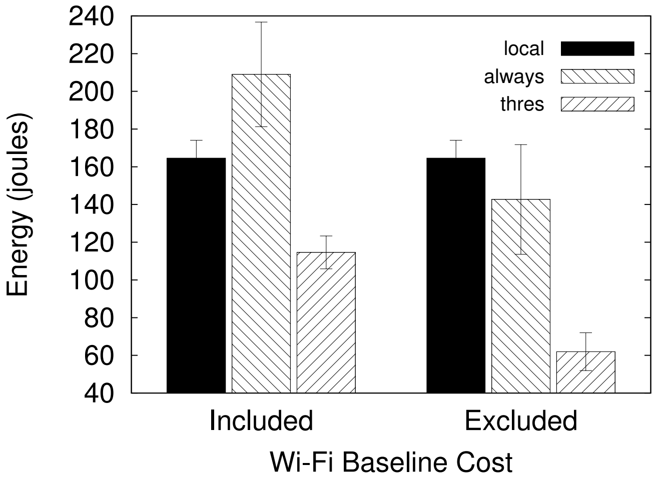


Figure 4.17: Energy consumption per hour with an inexpensive and an expensive sensor.

Multiple Sensors with Different Power and Data Requirements.

In this case, we evaluated the schemes using an inexpensive (30mW) and an expensive (1.2W) sensor. We assume that the inexpensive sensor generates a large amount of data per sensor sample (100KB per sample, similar to the audio recording for 5 seconds using a PCM format in the conversation detection module). Figure 4.17 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 scheme consumes 46% and 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% and 63% less energy than the always offload and always local schemes, respectively. The always local scheme consumes high energy because of local sensing of the expensive sensor, and the always offload scheme consumes high energy because of the communication cost of the inexpensive sensor. However, the energy consumption of threshold scheme is much lower than the other schemes, as it selects the optimal configuration for each sensing modality.

In this section we analysed the performance of the proposed threshold-based offloading scheme under a wide range of conditions. Based on these results, it is clear that there are situations where local sensing is preferable, while others where offloading is the most appropriate option. The threshold scheme tends to follow the optimal solution under all the conditions that we evaluated.

4.8 Related Work

As discussed in the related work of the previous chapter (Chapter 3, Section 3.6), several mobile phone based sensing systems [LYL$^+$10, RMM$^+$10b, WLA$^+$09] were proposed in the last few years. Several mobile sensing applications are presented in [CEL$^+$08]. Llama [BRC$^+$07] 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. However, most of these systems rely on local phone sensing, which is generally costly in terms of energy consumption as shown in this chapter. Comparing with these systems, the proposed system uses a novel sensing offloading approach for achieving energy efficiency, furthermore, as shown in the benchmarks evaluation, it can complement phone sensing systems using adaptive sampling to further improve the energy savings.

With respect to using the sensors in the infrastructure: *FollowMe* [GL11] lets mobile applications exploit the sensors in the existing infrastructure like cameras and microphones for richer context and better applications. They present some novel applications, for example, an application that creates an interactive video diary of a family's experience in a theme park using infrastructure sensors such as cameras placed near rides and food places in the park. ErdOS [VRC11] 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. The authors of [DFN$^+$09] created an interaction method that enables users to control and interact with content presented on public displays. In particular, they use Bluetooth names to interact with smart environments, *i.e.*, users are allowed to access content by changing their Bluetooth device names. They built many applications for use on their displays such as an interactive map service, accessing photos on Flickr (a photo management and sharing website), and videos on YouTube (a video-sharing website) that can be accessed using their interaction method. In [PH11], the authors propose that mobile phones can serve as data mules for sensor networks due to their ubiquity and show that opportunistic muling is suitable for office-based deployments. Even though this work involves interaction of mobile phones with sensor networks, it addresses a very different problem to ours. None of these works that exploit the sensing infrastructure provided a solution to the problem of *when to offload phone sensing to infrastructure*. Our technique provides an efficient way to support long-term deployment of social sensing applications by using a dynamic sensing offloading scheme, which switches between phone sensing and remote sensing considering the sensing parameters and the mobility patterns of users.

Some works [LFO$^+$07] deployed sensor networks at the workplace and domestic environments to understand usage patterns such as light use, sound, temperature, electrical current and voltage, and motion. These approaches rely on full instrumentation of a building that may again be costly or impractical.

4.9 Conclusions

In this chapter we presented a novel approach that can offer considerably bigger reductions in smartphone energy cost without compromising accuracy by *opportunistically offloading sensing to fixed sensors embedded in the environment.* The proposed scheme considers various parameters such as the mobility pattern of the user, duty cycling interval (or sensor sampling interval), cost of sensing on the phone to determine whether offloading sensing to the infrastructure results in energy gain at any given situation.

We conducted a study to explore the feasibility of offloading sensing and collected data to test the proposed offloading scheme. We showed through several benchmark tests on real traces that the scheme is able to achieve significant energy savings compared to pure phone sensing and remote sensing schemes. We also demonstrated that by combining the adaptive sensing techniques presented in the previous chapter with the sensing offloading scheme, the energy savings increase further.

5 Computation Offloading

5.1 Introduction

In the previous chapters we have discussed how adaptive sensing and sensing offloading schemes can be used to capture sensor data efficiently. Once data from the sensors is sampled, it needs to be processed to infer higher level activities. This processing might be negligible in terms of resource consumption for some classification tasks, for example, detecting whether a person is stationary or moving involves calculating the magnitude of acceleration and standard deviation (Chapter 3, Section 3.5), which are less intensive computing tasks. However, the processing requirements for some other classification tasks are high. Some examples are speaker identification from microphone data [LBBP+11] or image recognition from camera data [CBC+10]. Classification tasks are critical to the functioning of social sensing applications on mobile phones [LML+10]. Modern mobile phones are equipped with powerful processors, however, some classification tasks may need far higher processing power [KAH+12, CBC+10]. Moreover, using the local phone computing resources will consume energy and results in faster depletion of the phone battery. With the advent of cloud computing platforms such as Amazon Elastic Compute Cloud (EC2)[1], Windows Azure[2], and Google App Engine[3], computation tasks can be offloaded to the cloud for processing. Further, the accuracy of some classification tasks

[1]http://aws.amazon.com/ec2

[2]http://www.windowsazure.com/en-us

[3]https://developers.google.com/appengine

such as speech to text translation (e.g., Google voice search on the Android platform[4]) can be improved by using the "dictionaries" or other data available in the cloud. But, if the tasks are computed in the cloud, then the data needs to be transferred to the remote servers over a Wi-Fi/3G connection. For example, for audio processing the sound files may need to be transferred to the server/cloud and these files can be large. 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.

Our solution for computing classification tasks is to design a scheme that decides where to perform the computation, i.e., locally on the phone or remotely in the cloud by considering the various requirements of the user and experiment designers. In this chapter, we present the design of a computation distribution scheme based on *multi-criteria decision theory* [KR76] that decides where to perform the computations by considering various dimensions such as energy, latency, and data to be sent over the network. This scheme smartly distributes the classification tasks among local and cloud resources while balancing the energy-latency-traffic trade-offs. We also design a rule-based framework for dynamically adapting the behaviour of the scheme with respect to changes in mobile phone resources (like battery charge/discharge cycles, user's data plan running out of allowance). The computation offloading scheme is a step towards answering *Research Question 2 (How can we efficiently process data captured through smartphone sensors to draw inferences about the user?)* presented in Chapter 1.

Chapter outline. We present the offloading scheme and an XML based adaptation framework in Section 5.2. In Section 5.3, we present the design of an API that can be used by the social sensing applications to utilise the services of the proposed offloading scheme. The evaluation of the scheme through several micro-benchmark tests is presented in Section 5.4, and related work is presented in Section 5.5. Finally, we present conclusions in Section 5.6.

5.2 Computation Offloading

The computation offloading scheme is responsible for extracting high level inferences from the sensor raw data by processing the classification tasks efficiently using local phone and cloud resources. In this section, we present the design of the computation offloading scheme and a framework that enables it to adapt to the changing resources of the mobile phone.

[4]http://www.google.co.uk/intl/en_uk/mobile/voice-search/

5.2.1 Assumptions

In order to apply the computation offloading scheme, the following assumptions should hold.

- We refer to the process of classifying data from a sensor with respect to a classifier as a *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 the offloading module beforehand. This is a practical assumption as these values can be estimated for most of the common sensing tasks relatively easily: latency and the size of the data sent over the network can be easily estimated by the phone, and tools like the Nokia Energy Profiler [NEP] and the Monsoon Power Monitor can be used to obtain the energy values on the Symbian S60 platform and the Android platform, respectively.
- We also assume that classifiers are preloaded both locally on the phone and remotely in the server/cloud.
- We assume that a task takes constant time to compute, however, for some classifiers, the energy consumption and latency might vary based on the size of the input data. In these cases, the application should configure these parameters according to the size of the input data before executing the decision engine.

5.2.2 Method

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 characterising features from the audio file and processing them to identify the speaker. Similarly, a photo tagging task can be subdivided into scanning for the faces of people, feature extraction from each of these, and then comparison with existing models to perform the photo tagging.

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. For example, a speaker identification task (T) can be divided into two subtasks: extracting features from the recorded audio file (t_1), and comparing this with the existing models (t_2). Each subtask can be computed on the phone or in the cloud, therefore, it may have multiple versions, for example, subtask t_1 can be computed on the phone using subtask version v_{11} and in the cloud using subtask version v_{12}. We refer to each version of a subtask as a *subtask version*. If each subtask can be computed locally on the phone or remotely in the cloud then it results in a total of 2^n unique combinations of the breakdown of T into subtasks. Although the number of combinations is of exponential order, n is not the input size but the number of subtasks of task T, which in practice is small [CBC+10]. We discuss the complexity of the algorithm in detail towards the end of this subsection. 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* [KR76] 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:

$$u_{e_i} = \frac{e_{min} - e_i}{e_i} \tag{5.1}$$

where e_i is the energy consumption for processing task T using 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 [Fis68, KR76] 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 note that the range of the utility function ($[-1, 0]$) is negative because we model the utility of a configuration for a dimension (such as energy) as its gain with respect to the best performing configuration, which would be at most zero in the case where the configuration is the best (for example when $e_i = e_{min}$) with respect to the given dimension.

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.

$$u_{l_i} = \frac{l_{min} - l_i}{l_i} \tag{5.2}$$

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

$$u_{d_i} = \frac{d_{min} - d_i}{d_i} \tag{5.3}$$

where d_i is the size of the data sent over the network for processing the task T using the configuration c_i and d_{min} is the minimum of $\{d_i, i = 1, 2, \ldots 2^n\}$. The value of d_i is zero when computation is performed locally on the phone: In cases like this where the value of a dimension is zero, we also consider the value of the utility function to be zero. The decision about 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 [Fis65, Kee02] defined as follows:

$$u_{c_i} = w_e u_{e_i} + w_l u_{l_i} + w_d u_{d_i} \tag{5.4}$$

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 need not worry about the amount of data sent over the network, but will be concerned about the battery, so the experiment designers may give a higher weight to energy utility than to data utility. Finally, the configuration c_i for which u_{c_i} is maximum is used to process the task T.

This scheme can also be generalised to make it applicable to various other scenarios considering additional performance metrics or dimensions, and computation models (like cloud computation through Wi-Fi and 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:

$$u_{D_{ji}} = \frac{D_{jbest} - D_{ji}}{D_{ji}}, \forall\ j \text{ in } [1, 2, ..\ k],\ \forall\ i \text{ in } [1, 2, ..\ q^n] \tag{5.5}$$

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:

$$u_{c_i} = \sum_{j=1}^{k} w_j u_{D_{ji}},\ where \sum_{j=1}^{k} w_j = 1,\ \forall\ i \text{ in } [1, 2, ..\ q^n] \tag{5.6}$$

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. The scheme described in this section is shown in algorithmic format in Algorithm 1.

5.2.3 Algorithmic Complexity

The main computational tasks involved in this scheme are: (i) finding the best values for each of the dimensions (line number 7 in Algorithm 1) and (ii) computing the utility functions for each of the performance metrics (line number 17) and the total utility value for each of the configurations (line numbers 18). The algorithmic complexity of both these procedures (line numbers 4 to 10, and 15 to 24) is $O(kq^n)$, where n is the total number of subtasks that a given task T can be divided into, k is the total number of dimensions, which is 3 in our case (energy, latency, and data sent over the network), and q is 2 as we consider local and remote computation models. Since these procedures are executed one after the other sequentially and not inside a loop, the overall algorithmic complexity is $O(2^n)$. To find the best combination, the utility value for each of the combinations has to be evaluated. Since this requires iterating through all the combinations, it results in an exponential complexity. Even though this is exponential, the total number of subtasks n

```
<rules>
    <condition expr="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 expr="battery_left < 20">
       <weight metric="energy">60</weight>
       <weight metric="latency">20</weight>
       <weight metric="data">20</weight>
    </condition>
    <condition expr="data_sent > 50MB and data_sent < 100MB">
       <weight metric="energy">20</weight>
       <weight metric="latency">20</weight>
       <weight metric="data">60</weight>
    </condition>
    <condition expr="data_sent > 100MB">
       <weight metric="energy">0</weight>
       <weight metric="latency">0</weight>
       <weight metric="data">100</weight>
    </condition>
    <condition expr="default">
     <weight metric="energy">33.3</weight>
     <weight metric="latency">33.3</weight>
     <weight metric="data">33.3</weight>
    </condition>
</rules>
```

Figure 5.1: Sample rules for adaptation of weights for energy, latency, and data sent over the network. If none of the conditions match the current status of the phone then the weights specified in the default condition will be used.

is, in practice, small [CBC+10, MCR+10]. For example, in the case of computationally intensive tasks like speaker recognition n is generally around 2 to 4: 1) feature extraction such as amplitude, Fast Fourier Transforms (FFTs) etc., 2) detect if the features represent voice data, and 3) compare the extracted features with existing models such as speaker models or emotion models [MCR+10, RMM+10b, LRC+12].

Algorithm 1 Computation offloading decision algorithm

```
1: **Input:** A task T with $n$ subtasks
2: **Output:** SubTaskVersion[ ] subTaskVersionArray
3: Configuration[ ] configuration = buildConfigurations(T)
   {Each configuration represents a combination of subtask versions of T and has details
   about all the dimension values for this combination of subtask versions.}
4: Dbest[j] $\leftarrow +\infty$, $\forall\, j$ in $[1, 2, ..\, k]$
   {Dbest[j] is the best value for dimension j across the configurations}
   {Calculate Dbest[j] for all dimensions}
5: for i = 1 to $q^n$ do
6:    for j = 1 to k do
7:       if configuration[i].getDimensionValue(j) < Dbest[j] then
8:          Dbest[j] = configuration[i].getDimensionValue(j)
9:       end if
10:   end for
11: end for
12: Configuration bestConfig $\leftarrow$ *null*
13: Ubest $\leftarrow -\infty$
14: u[i] $\leftarrow 0$, $\forall\, i$ in $[1, 2, ..\, q^n]$
    {u[i] is the utility of configuration i}
15: uD[i][j] $\leftarrow 0$, $\forall\, j$ in $[1, 2, ..\, k]$, $\forall\, i$ in $[1, 2, ..\, q^n]$
    {uD[i][j] is the utility of $j^{th}$ dimension of $i^{th}$ configuration}
16: w[j] $\leftarrow$ loadWeightFromXMLConfig(), $\forall\, j$ in $[1, 2, ..\, k]$
    {w[j] is the weight for $j^{th}$ dimension}
    {Calculate utilities for all the configurations and find the best configuration}
17: for i = 1 to $q^n$ do
18:    for j = 1 to k do
19:       uD[i][j] $\leftarrow \frac{\text{Dbest[j] - D[i][j]}}{\text{D[i][j]}}$
20:       u[i] $\leftarrow$ u[i] + (w[j] * uD[i][j])
21:    end for
22:    if u[i] > Ubest then
23:       Ubest $\leftarrow$ u[i]
24:       bestConfig $\leftarrow$ configuration[i]
25:    end if
26: end for
27: subTaskVersionArray $\leftarrow$ bestConfig.getSubTaskVersions()
    {return the subtask versions in the configuration with highest utility}
28: return subTaskVersionArray
```

5.2.4 Adaptation of Weights

The users of the social sensing systems that use the proposed offloading scheme are expected to provide the weights for energy, latency, and total data sent over the network based on the requirements of the participants of social sensing studies. However, unlike other computing devices like desktop computers, the resources of mobile phones are not static and some resources like battery life and total data left in a user's data plan change over time. The battery charge of mobile phones lasts for a limited amount of time, 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 "resource states" of the devices. For example, the users 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. 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 metrics as resource levels change. The specification of rules is in XML format, and a sample configuration is shown in Figure 5.1. Although we chose XML as a format for configuration of the rules, other data exchange standards such as JavaScript Object Notation (JSON) could also be used. The weights configured for the condition that matches first will be used, which also addresses the case of defining conflicting rules. If none of the conditions are satisfied then the weights configured as default will be used. In the sample configuration given in the figure, the weights are distributed equally when the battery level (`battery_left`) is high and data plan consumption on the current day (`data_sent`) is below 50MB, however, when the battery level falls below 20%, the weight for energy consumption is increased to 60%. Similarly, when the data sent over the network (on the current day) crosses 50MB but is below 100MB then the weight for data is increased to 60%, and if the data sent crosses over 100MB, then all the weight is given to the data traffic.

5.3 Computation Offloading API

We designed an API for the computation distribution scheme, which can be used by mobile applications to use the services of the computation offloading scheme to efficiently utilise the local phone and remote computing resources. As discussed in the previous section, each classification task can be divided into one or more subtasks and each subtask can have one or more subtask versions. In designing the computation offloading API, first, we define an XML schema that can be used to represent tasks and then an API to use the offloading scheme.

5.3.1 Definitions

In this subsection we describe the XML schema that can be used to define tasks.

- **Task.** Task represents a classification task such as “speaker identification” or “physical activity classification”.
- **Subtask.** A task is divided into subtasks, which have to be executed sequentially to complete the task. Subtasks of “speaker identification” could be “extracting features” and “comparison with the speaker models”. The output of a task is used as the input for the next task.
- **Subtask Version.** Each subtask can have several subtask versions implementing the subtask. For example, a subtask version can run locally and another subtask version can run remotely. Each subtask version has a certain latency, energy, a computation model (local or remote), expected input data size, and expected output data size.

Based on the requirements of the task and the weights given to the dimensions: energy, latency, and data traffic (as described in the previous section), the decision engine decides which subtask version to execute for each subtask. The programmer of a task can specify the properties and requirements either in an XML configuration file or programmatically using the API.

5.3.2 XML Task Specification

In this section we describe the XML format of the file in which tasks are specified. The tasks can also be specified using the API methods. We present a detailed description of the API and the XML format in Appendix B. The developer creates the XML file either manually or with the help of a task profiler. The file is stored in the directory `/res/xml/`[5] of an Android application project.

The format of the file is as follows:

- **Element: task.** The task element has a string attribute “id” which defines the identifier of the task. The task element contains one or more subtask elements. It contains elements defining the pre-filtering requirements for the task in terms of privacy and maximum latency. The pre-filtering is used to filter the configurations that do not meet the specified requirements (privacy, latency) from the offloading decision process.

[5]Android `res` folder contains application resources such as custom configuration files, layout files, and string values. http://developer.android.com/guide/topics/resources/index.html

```xml
<?xml version="1.0" encoding="UTF-8"?>
<tasks xmlns="Cambridge:CO"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="Cambridge:CO tasks.xsd">
    <task id="speech_recognition">
        <max-latency>100000</max-latency>
        <min-privacy-level>low</min-privacy-level>
        <subtask id="extract_features" position="1">
            <subtask-version version="1">
                <computation-model>remote</computation-model>
                <input-size>2048</input-size>
                <output-size>128</output-size>
                <energy>10</energy>
                <latency>500</latency>
            </subtask-version>
            <subtask-version version="2">
                <computation-model>local</computation-model>
                ....
            </subtask-version>
        </subtask>
        <subtask id="compare_with_models" position="2">
                ....
        </subtask>
    </task>
</tasks>
```

Figure 5.2: Sample specification of tasks.

- **Element: max-latency.** Contains a long value of the maximum allowable latency for any subtask-version.
- **Element: min-privacy-level.** Specifies the privacy level of input and output data for a subtask. Value can be either high (data may not be sent over the network) or low (data may be sent over the network). If the value of "min privacy level" is high, then all the processing is performed locally on the phone, otherwise, the processing is performed as per the decision of the offloading scheme. If the user is concerned about transferring his data to remote servers for processing, then he could disable this by setting the value of "min privacy level" to "high".

- **Element: subtask.** The subtask element has a string attribute “id” which defines the identifier of the subtask. Further, it has an integer attribute “position” which defines the position in the sequence in which the subtasks are executed. It contains one or more subtask-version elements.
- **Element: subtask-version.** The subtask-version element contains an attribute “version”, a string identifier for the subtask version. Further, it contains elements specifying the properties of the subtask version: energy, latency, input size, output size, and computation model (local, remote).
- **Element: energy.** Contains a long value of the expected energy consumption (in Joules) of the subtask version.
- **Element: latency.** Contains a long value of the expected latency (in milliseconds) of the subtask version.
- **Element: input-size.** Contains an integer value with the expected input data size in bytes.
- **Element: output-size.** Contains an integer value with the expected output data size in bytes.
- **Element: computation-model.** ENUM: local, remote

An example XML file is shown in Figure 5.2 for a speaker identification task, which consists of two subtasks: extracting features and model comparison, each of which has several versions (local or remote). The sequence of operations that need to be performed by developers to use the computation offloading API is shown in Figure 5.3. We present more details of the API in Appendix B.

5.4 Micro-benchmarks

In this section, we present the evaluation of the computation distribution scheme with respect to selecting the best suitable configuration based on real traces collected through participants carrying mobile phones.

5.4.1 Empirical Datasets

The dataset used for evaluating the computation offloading scheme is based on that used in Chapter 3 (Section 3.5.1) for evaluating adaptive sensing schemes. We note that since the computation offloading scheme do not need support from the sensing infrastructure

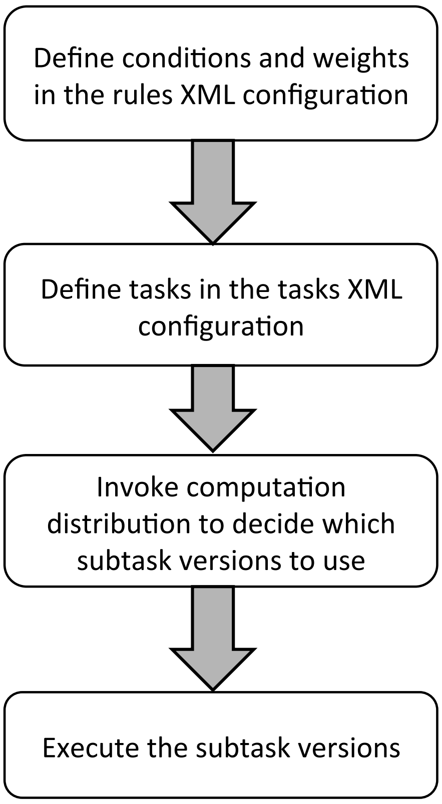

Define conditions and weights in the rules XML configuration
Define tasks in the tasks XML configuration
Invoke computation distribution to decide which subtask versions to use
Execute the subtask versions

Figure 5.3: Sequence of operations to be performed to use the computation offloading API.

(remote sensing, indoor localisation etc.) we did not use the dataset described in the previous chapter (Chapter 4, Section 4.5), instead we used the dataset described in the adaptive sensing chapter that contains data collected from the phone's sensors. We briefly describe it here. The dataset was collected by 10 users carrying a Samsung Galaxy S or Nokia 6210 Navigator phone within our research institution (the Computer Laboratory, University of Cambridge). The data collection application collected raw accelerometer sensor data (i.e., X, Y, Z coordinates), Bluetooth data (i.e., Bluetooth identifiers), and microphone data (i.e., audio recordings). The sampling of the accelerometer and Bluetooth sensors was performed continuously with a sleep interval of 0.5 seconds and 1 second, respectively. Audio samples of length five seconds were recorded from the microphone sensor with a sleep interval of one second between consecutive audio recordings. Samsung Galaxy S phones were running *Google Android version 2.1* [AND] or higher, and Nokia 6210 Navigator phones were running *Symbian S60*. The choice of operating system does not impact the data collection process as the duty cycling rates used were same in both the phone models. The raw data in the collected dataset was then classified into high level events using the classifiers described in Chapter 3, Section 3.5.2. We used the adaptive sensing scheme described in the same chapter as sensor sampling scheme for the evaluation of computation offloading scheme.

5.4.2 Method

The main aim of these benchmarks is to evaluate the performance of the utility function (explained in Section 5.2) used in the computation distribution scheme in terms of selecting the best configuration given the importance assigned to each performance dimension. More specifically, we compared the performance of all possible configurations and evaluated 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.** The total time until battery is completely discharged from the fully charged state.
- **Average latency.** The average time taken for processing a classification task.
- **Average data sent over the network.** The average number of bytes sent by the system over the network to process a sensor classification task.

5.4.3 Tasks used in the Benchmarks

The classification tasks used in the benchmarks are of three types based on the sensor from which data is queried.

- The *activity recognition* classification task has one subtask that classifies the data sensed from the accelerometer sensor into *moving* or *idle* states. The computation in this task is less intensive and involves calculation of magnitude of acceleration using the [x, y, z] vectors, and then the standard deviation of the magnitude values. The standard deviation is compared with a threshold value to derive the physical activity state of the user. The procedure is detailed in Chapter 3, Section 3.5.2.
- The *co-location detection* classification task has one subtask that detects the change in colocation of the user observed through the change in co-located Bluetooth devices. The computation involved in this task is also less intensive and involves computing the difference between two sets of Bluetooth device addresses.
- The *speaker identification* classification task detects the speaker. This task is highly intensive and contains two subtasks.
  1. 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) [HTK], which is an intensive computation.

2. The second subtask compares the extracted coefficients file with the speaker models of all the users using the *HERest* tool of HTK. The model with the highest likelihood of match is selected as the speaker model of the recorded audio file.

The design and implementation of these tasks will be explained in detail in the next chapter (Chapter 6, Section 6.2.2). This is also an intensive computation, in which the duration of computation linearly increases with the number of speaker models available for comparison.

Each of these tasks have two subtask versions to perform computation of the task either locally on the phone or remotely in the cloud.

5.4.4 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:*** both subtasks computed locally.
- ***C2:*** first subtask computed locally and the other remotely.
- ***C3:*** first subtask computed remotely and the other locally.
- ***C4:*** both 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 to all the dimensions). These are shown in Table 5.1. Since our main aim is to show that the utility function selects the best configuration considering the weights, we limit the test combinations to these four combinations that represent common use cases, i.e., providing all the weight to a single dimension or dividing the weight equally to all the dimensions.

Figures 5.4, 5.5 and 5.6 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 combination *S1* selects configuration *C4* as this has the lowest energy consumption, and for combination

Combination	W_e	W_l	W_d
S1	1	0	0
S2	0	1	0
S3	0	0	1
S4	0.33	0.33	0.33

Table 5.1: Weights for energy, latency, and data traffic for various combinations used in the evaluation.

S2 the configuration selected is *C4* as this is the lowest in terms of latency as well. For combination *S3*, configuration *C1* is selected as this sends the least data over the network. Finally, for combination *S4*, which gives equal weights to all the dimensions, 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 (configurations C4, C4, C1, C4 for combinations S1, S2, S3, S4, respectively) 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 the 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 considered 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. *C1* computes all the subtasks locally on the phone, *C2* computes the second subtask of the microphone task remotely and all others locally, and so on, *C15* computes the second subtask of the microphone task locally and other subtasks remotely, and finally, *C16* computes all subtasks remotely in the cloud. We evaluate the utility function for the same combination of weights: *S1*, *S2*, *S3*, and *S4*. The learning based technique presented in Chapter 3 is used as the sensor sampling mechanism.

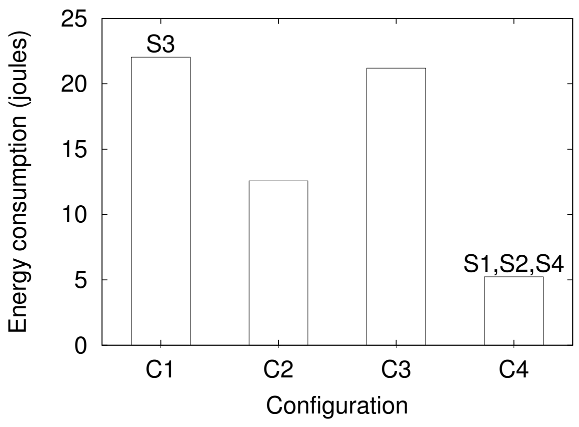


Figure 5.4: Energy consumption for processing the speaker identification task.

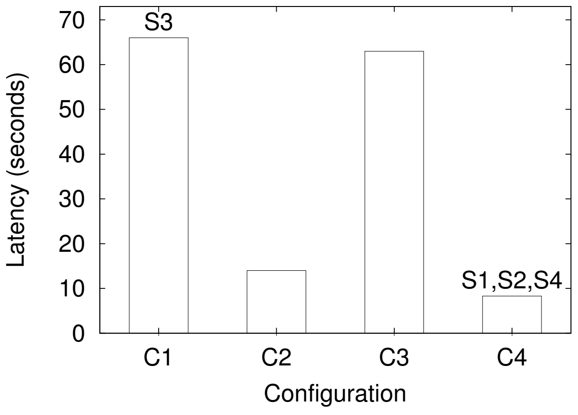


Figure 5.5: Latency or delay for processing the speaker identification task.

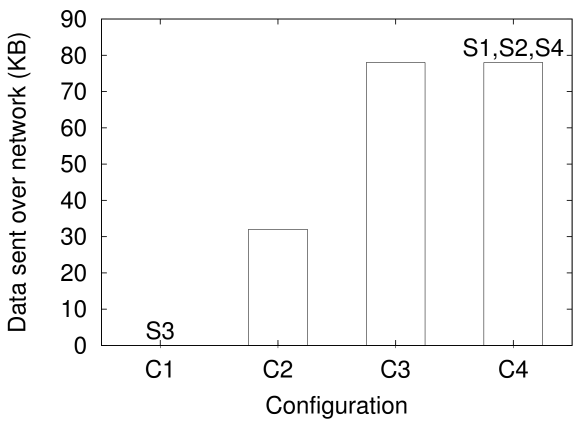


Figure 5.6: Data sent over the network for processing the speaker identification task.

Figures 5.7, 5.8 and 5.9 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 combination *S1* that gives maximum weight to energy, the configuration with the highest overall lifetime is selected (i.e., *C4*) . For combination *S2* that gives maximum weight to latency, configuration *C4* is selected as this is the lowest in terms of latency as well. For combination *S3*, the configuration with least 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*, and *C13*, and from them the configuration with better latency and lifetime values (i.e., *C1*) is selected. Finally, for combination *S4*, configuration *C4* is selected as it is better in two out of three dimensions (lifetime and latency) 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.

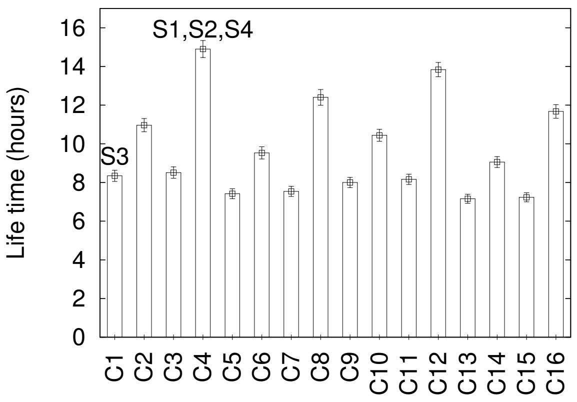


Figure 5.7: Total lifetime of the mobile phone.

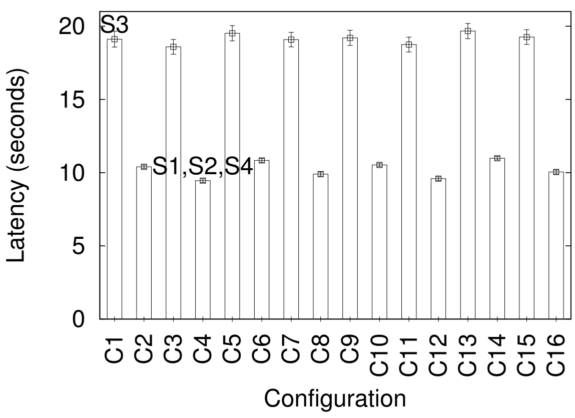


Figure 5.8: Average latency of processing a task.

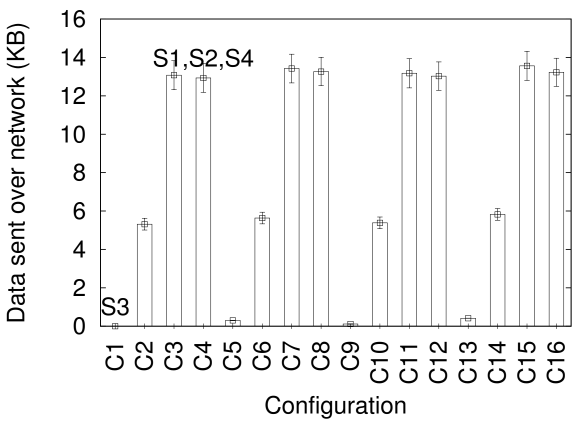


Figure 5.9: Average data sent over the network for processing a task.

5.5 Related Work

There have been much work focussed on increasing the efficiency of mobile systems by offloading computations [CBC+10, KAH+12, CIM+11, BSPO03]. In [MPF+10], 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 minimisation of the amount of data sent to the back-end servers and not on the distribution or offloading of the computation. The authors of [RNRR10] present the design and implementation of *NAPman*, a network assisted power management scheme for Wi-Fi equipped smartphones. It employs an energy-aware fair scheduling algorithm (at the Access Point) to reduce Wi-Fi energy consumption of mobile clients and eliminates unnecessary retransmissions. In [KL10], the authors discuss the energy savings when computation is offloaded to the cloud and present aspects related to privacy and reliability of mobile systems such as smartphones.

Cool-Tether [SNR+09] provides energy-efficient and affordable connectivity to the cloud by building a Wi-Fi hotspot utilising the cellular radio links of one or more smartphones nearby. These approaches are orthogonal to our offloading scheme, and the proposed scheme can work alongside these techniques.

Spectra [FPS02] is a remote execution system for mobile devices that chooses the execution plan (provided by the programmers) dynamically based on performance, energy, and application quality. In Spectra, mobile applications statically specify which code components benefit from remote execution. During the execution, the system monitors resource usage and advises on how and where to execute the specified code components. This helps mobile applications to adapt to resource changes without specifying their resource requirements. However, remote execution requires data to be transmitted to remote servers and most mobile data plans have limits on how much data can used per day or per month. The costs after exceeding these limits are generally high. Spectra does not consider the data plan limits nor it considers adapting the offloading scheme according to the data usage. Further, it lacks a way of dynamically adapting the priorities (or weights) given to dimensions with respect to the changing resources of mobile phone, whereas we consider this in the proposed offloading scheme (through dynamic adaptation using rules specified by the user).

CloneCloud [CIM+11] enables mobile applications running in an application layer virtual machine (such as the Java Virtual Machine (JVM)[6], the Dalvik Virtual Machine on the Android Platform, and the Microsoft's .NET platform[7]) to offload part of the execution to device clones running in the cloud to make the execute time faster and also energy efficient. In this scheme, at runtime, an application thread is migrated at a chosen point from the mobile phone to the device clone in the cloud and then computed. It is then re-integrated back to the phone. The system is mainly useful for mobile applications running in a virtual machine.

In [KPKB10], the authors present a framework for supporting computation offloading on the Android platform. They show using examples such as image recognition and an augmented reality game that computation offloading of intensive tasks using the proposed framework is beneficial. However, they do not address the problem of whether to offload a computation task or not. They mention that the framework always offloads a task to the cloud if there is an opportunity. Scavenger [Kri10] is also a framework for supporting computation offloading and utilises the cross-platform support of the Python platform. They use a scheduler to estimate the cost of performing computation remotely by considering parameters such as speed of the remote machines, network bandwidth and latency, data size etc. However, they do not consider the energy consumption of performing these tasks. Further, our scheme considers the requirements of the users and the changing re-

[6]http://www.java.com/en/

[7]http://www.microsoft.com/net

sources of the phone (such as the phone's battery level), whereas the Scavenger system does not support these features.

ThinkAir [KAH+12] is a framework for offloading mobile computation to the cloud. The system accommodates changing computational requirements based on on-demand VM resource scaling, and exploits parallelism for faster execution. The authors of [CBC+10] present MAUI, a system that achieves energy efficiency through fine-grained code offloading to the infrastructure while minimising the required code-level changes to applications. They showed that MAUI considerably improves the energy consumption of resource-intensive applications, for example, they showed that by using code offloading MAUI saves 27% energy for a video game. Furthermore, they also showed that the energy savings are much higher for a face recognition task, and its latency is reduced from 19 seconds to less than 2 seconds by exploiting remote computation. With respect to these works, our scheme provides a dynamic decision engine that decides where to perform the computation of classification tasks i.e., locally on the phone or remotely in the cloud considering the requirements of experiment designers (like social scientists) in terms of battery consumption, delay, and traffic trade-offs. Moreover, the proposed offloading scheme implements mechanisms that are adaptive to the user's and designer's requirements (battery, data plan, etc.) while considering dimensions such as energy, latency, and data traffic.

In [LPH+12], the authors present Cloud-Offloaded GPS (CO-GPS) that allows a mobile phone to duty cycle its GPS receiver and capture just enough raw GPS signal for post-processing in the cloud by using publicly available satellite ephemeris information. The authors show that, by utilising the cloud processing, 2 milliseconds of raw GPS signals is enough to obtain a location fix. CO-GPS addresses the problem of offloading GPS raw data to the cloud, and does not consider the problem of whether to offload a computation to the cloud or not, which is addressed in the current chapter.

5.6 Conclusions

Social sensing applications use classifiers to infer various social and behavioural aspects of users from raw data captured by smartphone sensors. The computing requirements of these classifiers may vary from trivial to highly intensive depending on the type of classification task.

In this chapter we presented a computation offloading scheme that efficiently utilises local phone and remote cloud computing resources to perform the computation of classification tasks. The core of the component is a decision engine that decides whether to perform the computation for a given task locally on the phone or remotely in the cloud by considering the dimensions: energy, latency, and data sent over the network. We also described an adaptation framework for dynamically adapting the scheme to the changing resources of the phone based on an XML configuration file. We then presented the design of an API, which can be used by mobile applications to utilise the services of the offloading scheme. We showed through evaluation using real traces that the offloading decisions taken by the scheme are optimal, and that our scheme dynamically selects the optimal computation model according to the given requirements.

The main limitations of the computation offloading scheme are:

1. The complexity of the algorithm (presented in Section 5.2, Algorithm 1) is $O(2^n)$, where n is number of subtasks in a given task T. We, however, note that n, in practice, is small [MCR+10, LRC+12].

2. The rules framework (presented in Section 5.2.4) to configure the weights for the dimensions does not provide a mechanism to validate the rules, for example, to check for conflicting rules. The main focus of the work in this chapter is on deciding where to perform the computation of classification tasks, however, a future work could be to provide an interface to write and verify the rules.

6
Social Sensing Applications

6.1 Introduction

In the previous chapters we have discussed techniques that adaptively sample a smartphone's sensors to balance energy-accuracy trade-offs, improve energy efficiency by exploiting sensors in the user's environment, and efficiently compute classification tasks using the phone's and cloud resources while considering the user's requirements. Efficient sensing and processing on smartphones finds application in many domains such as environmental noise monitoring (using the phone's microphone), physical health and well-being (using the phone's accelerometer), systems requiring indoor localisation such as co-location monitoring at the workplace (using the phone's Bluetooth radio), and social sensing (using combinations of sensors). Social sensing is useful in many domains such as mental health, quantified self applications, and social and behavioural psychology. We choose social psychology as the main application domain to apply our techniques as it deals with a diverse set of social sensing aspects such as interactions, emotions, behaviour, and the influence of various factors such as physical activity, location, co-location on these aspects. The idea is to develop a set of tools and techniques, so that the developers interested in building applications for the social psychology domain can focus on the application development while depending on the techniques proposed in this dissertation for achieving energy-efficiency. Further, by targeting a large domain such as social psychology, the potential applications that could be developed using the proposed tools can be much more compared to the case of targeting a specific application scenario.

In this chapter, we design and deploy various example social psychology applications while demonstrating that the proposed techniques assist in performing efficient social sensing. In particular, we answer *Research Question 3* (*In what ways can smartphones be helpful in the conduct of social studies*) presented in Chapter 1, by showing that smartphones can be used as a platform for building social psychological applications such as capturing the behaviour of individuals in terms of speech and emotional patterns, monitoring workplace behaviour and identifying collaboration and work patterns, and providing feedback to help users in fostering their interaction. For this, we design various techniques such as speaker identification, emotion detection, detecting strength of relations, and detecting collaborations. Using these techniques, we design example social psychological applications to demonstrate the type of data that can be collected using smartphones and their sensors and the analysis that can be performed on this data. These applications are deployed as part of social sensing systems built using the services of the schemes presented in the previous chapters. Our aim in building these applications is to show some examples of how smartphones can be useful in social psychological deployments and to further demonstrate that the techniques presented in the previous chapters help in reducing the energy consumption.

Systems such as CenceMe [MLF^{+}08] and Betelgeuse [KLNA09] have shown the potential of mobile phone sensing in providing information such as user movement and activity for recreational and healthcare applications. As discussed in Chapter 1, one possible use of these technologies is arguably the support of sociology experiments [MGP06] which involve studying people's daily life and interaction. In the past, this analysis has been performed with the help of cameras (in home/working environments or in laboratories), use of voice recorders attached to people [MP01], and self reports using daily diaries or PDAs [BDR03]. However, these techniques may lead to biased results since people are aware of being constantly monitored [PR91]. Instead, mobile phones may offer an *unobtrusive* means of obtaining unbiased information about the behaviour of individuals and their interaction. However, techniques to accurately infer the user's behaviour from raw data of sensors need to be developed. In this chapter, we design classifiers for behaviour modelling of users and present the design, implementation, and deployment of three example social sensing applications based on smartphones.

Chapter outline. In Section 6.2, we present our first example application, EmotionSense, a passive behavioural monitoring application that infers the user's emotions and speech patterns. We then present WorkSense in Section 6.3, another example application that captures the interaction and collaboration patterns of users at the workplace. SociableSense, an application that provides feedback to users for fostering interaction at the workplace is presented in Section 6.4. We discuss related work in Section 6.5 and finally, offer conclusions in Section 6.6.

6.2 EmotionSense

In this section, we present our first example application, EmotionSense, an application for collecting data in human interaction studies based on mobile phones. EmotionSense aims to infer participants' emotions as well as proximity and patterns of conversation by processing outputs from the sensors of off-the-shelf smartphones. This can be used to understand the correlation and the impact of interactions and activities on the behaviour of individuals. In particular, through EmotionSense, we show that smartphones could be potentially used to collect behavioural data of users and to understand the impact of various external factors such as activity, co-location on their behaviour. More specifically, in this section, we present the following:

- The design, implementation, and evaluation of an example application that could be used in social studies and is able to provide information inferred about user behaviour, especially with respect to the influence of activity, group interaction, and time of day.
- We show that offloading computation helps in reducing the phone's energy consumption for high intensive classification tasks such as speaker identification.
- The results of a real deployment to demonstrate the type of sensor data that could be collected by the application using off-the-shelf smartphones.

We note that the system does not store voice recordings and they are discarded immediately after processing. Bluetooth names are also not stored by the system.

6.2.1 Social Sensing

What are the typical emotions exhibited by people? What is the correlation of emotion with location and activity or with interaction? How do speech patterns vary between members of a group of users? Generations of social psychologists have tried to answer these questions using a variety of techniques and methods involving experiments on people.

Most social scientists rely on self-reports or behavioural observations of participants in laboratory settings, however, these methods are found to be biased [FMPP07, PR91]. As discussed in Chapter 1, mobile phone sensing technology has the potential to bring a new perspective to the design of social studies, both in terms of sensor data that could be collected and from a practical point of view. Mobile phones are already part of people's daily lives, so their presence is likely to be "forgotten" by users, leading to observation of spontaneous behaviour. The goal of EmotionSense is to exploit mobile sensing technology to collect social and behavioural data of users.

6.2.2 Application Components

In this subsection we provide details of the implementation of the fundamental components of EmotionSense, describing the key design choices and solutions. The speaker recognition component is implemented in C++ since it is based on tools of the Hidden Markov Model Toolkit (HTK) suite for speech processing, which was originally written in that language [HTK]. In the rest of the subsection we provide details of the core components of the EmotionSense system: speaker and emotion recognition.

Speaker Recognition

The speaker recognition component is based on a Gaussian Mixture Model classifier [Bis06, Rey02], which is implemented using HTK [HTK][1]. HTK is a portable toolkit for building and manipulating Hidden Markov Models (HMMs) and Gaussian Mixture Models (GMMs) and provides sophisticated facilities for speech analysis, model training, testing, and results analysis. At present HTK is available for Windows and Linux systems only. It was therefore necessary to adapt the main components of the toolkit to work on the Nokia Symbian S60 platform.

The speaker recognition process is performed as follows:

- Speech data are collected from all users enrolled in the current experimental study. The data are then parameterised using a frame rate of 10ms and a window size of 30ms and a vector of 32 Perceptual Linear Predictive (PLP) coefficients [Her90] (16 static and 16 delta) are extracted from each frame.
- A 128-component universal background GMM (representative of all speakers) is then trained using all available enrolment speech to optimise a maximum likelihood criterion. This training procedure is currently executed offline. However, the training procedure could also be executed at run-time by sending samples to the back-end servers by means of a WiFi or 3G connection.
- Next, a set of user-dependent GMMs are obtained by performing Maximum A Posteriori (MAP) adaptation of the background model using the enrolment data associated with each user. The adaptation constant used for the MAP process was set to 15.
- Finally, at run-time the likelihood of each audio sequence is calculated given each user model. Each sequence is then associated with the model that assigns it the

[1]Alternative SVM-based schemes, including the popular GMM-supervector [CSR06] and MLLR [SFK+05] kernel classifiers, were not considered as they are generally suitable for binary classification tasks.

highest likelihood. This is the Bayes decision rule [Bis06] in the case that the prior probability associated with each user is equal.

In order to improve the accuracy and efficiency of the system, two key mechanisms were implemented:

- *Silence detection.* Successfully detecting silence can improve the efficiency of the system by eliminating the need to compare each sequence with each user-dependent model. Silence detection was implemented by training an additional GMM using silence audio data recorded under similar background conditions to the enrolment data. Each audio sequence is initially classified as either silence or non-silence by comparing the likelihood of the sequence given the silence and the background GMMs. The silence detector can also be used to infer information about the user's environment, sleep patterns and so on.

- *Comparisons driven by co-location information.* To reduce the total number of comparisons required, the speaker recognition component compares a recorded audio sequence only with the models associated with co-located users. Co-location is inferred from data captured by the Bluetooth sensor. This avoids unnecessary comparisons against models of people who are not in proximity to the user, both considerably speeding up the detection process and potentially avoiding misclassifying the sequence as belonging to users who are not present.

Emotion Recognition

The emotion recognition component is also based on a GMM classifier. The classifier was trained using emotional speech taken from the Emotional Prosody Speech and Transcripts library [LDG+02], the standard benchmark library in emotion and speech processing research. This corpus contains recordings of professional actors reading a series of semantically neutral utterances (dates and numbers) spanning fourteen distinct emotional categories. The selection is based on Banse and Scherer's study [BS96] of vocal emotional expression. Actor participants were provided with descriptions of each emotional context, including situational examples adapted from those used in the original study. Flashcards were used to display series of four-syllable dates and numbers to be uttered in the appropriate emotional category.

The emotion recognition process is performed as follows:

- A 128-component background GMM representative of all emotional speech is initially trained using all the emotion data.

- MAP adaptation of the background model is performed offline using emotion specific utterances from the emotion database to obtain a set of emotion-dependent models. These models are then loaded onto the phones.

Broad emotion	**Narrow emotions**
Happy	Elation, Interest, Happy
Sad	Sadness
Fear	Panic
Anger	Disgust, Dominant, Hot anger
Neutral	Neutral normal, Neutral conversation, Neutral distant, Neutral tete, Boredom, Passive

Table 6.1: Emotion clustering

- At run-time, the component periodically calculates the likelihood of the recorded audio sequence given each emotion-dependent model and assigns the sequence to the emotion characterised by the highest likelihood.

We initially tested a total of 14 "narrow" emotions based on the classes defined in the emotion library. These were then clustered into 5 standard broader emotion groups generally used by social psychologists [FBR98]. It is difficult to distinguish with high accuracy between utterances related to emotions in the same group, given their similarity. In any case, we also note that it is also hard for a person involved in an experiment to distinguish exactly between the emotions belonging to the same group in a questionnaire and for this reason broad classes are commonly used. The details of each grouping are given in Table 6.1.

6.2.3 Evaluation

EmotionSense needs to be evaluated to optimise the speaker and emotion recognition components and to understand their accuracy and energy requirements. It also needs to be evaluated through a user study to understand the type of data that could be collected by it. We also make a case for the use of computation offloading scheme presented in Chapter 5.

In this subsection we present an evaluation of EmotionSense by means of several micro-benchmark tests to study the system performance and to tune the parameters of the speaker and emotion components. In particular, we discuss the choice of the optimal audio sample length value for speaker and emotion recognition. We also present an evaluation to understand the benefits of computation offloading. In the next subsection, we describe the results of a larger-scale deployment involving 18 users for 10 days to evaluate the prototype in a realistic setting and demonstrate its usefulness in the conduct of social studies. The evaluation was performed using Nokia 6210 Navigator phones. If we had used Android phones, it would not have affected the accuracy of results that will be presented in this section as the classification accuracy is independent of the phone's operating system, but would affect the latency if the Android phone had a more powerful processor.

6.2.4 Performance Benchmarks

We ran a series of micro-benchmarks to test the performance of the speaker and emotion recognition components. The data used for benchmarking was collected from 10 users. Each user carried a Nokia 6210 Navigator mobile phone. We also explored the trade-offs in performing local computation on the phones and remote computation on a back-end server.

Speaker Recognition

In this subsection, we present the results of the speaker recognition subsystem benchmarks. Voice samples from 10 users were used in this test. The users were students, staff, and faculty of the Computer Laboratory, University of Cambridge. The users that we considered have diverse cultural backgrounds and their nationalities are distributed among 7 different countries and 3 different continents. We used about 10 minutes of data from each user to train the speaker-dependent models. A separate, held-out dataset was used to test the accuracy of the speaker recognition component. We varied the sample length from 1 to 15 seconds and each sample was classified against 14 possible models. We used 15 samples per user per sample length, resulting in a total of 150 test samples per sample length. Figure 6.1 shows speaker recognition accuracy with respect to sample length. As sample length was increased, accuracy improved, converging at around 90% for sample lengths greater than 4 seconds. From Figure 6.2, it can be seen that this corresponds to a latency of 55 seconds in the case of local computation on the phone. In our tests, we observed a higher accuracy for silence detection than speaker identification. This is because, the speaker identification technique has been tested against 14 models, whereas for silence detection, the classification test is binary, i.e., comparison against silence and background GMM models. Further, it may also be because, it is relatively less complex for the classification process to differentiate between a sound signal and a silence signal compared to two sound signals with varying characteristics.

Computation Offloading

In this subsection, we show the advantages of offloading computation tasks to the cloud. We compared the energy consumption and latency of classifying each audio sample locally on the phone with that of remote classification on a powerful server. In the case of local

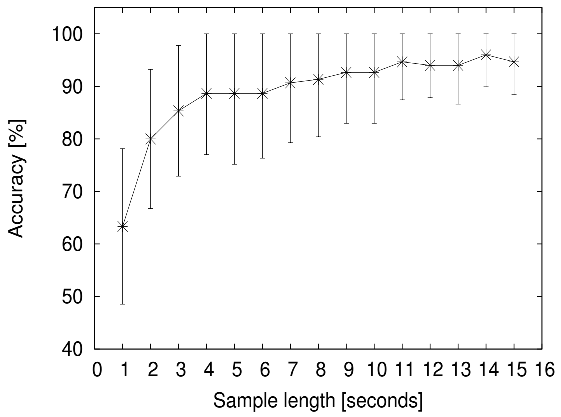


Figure 6.1: Speaker recognition accuracy vs audio sample length.

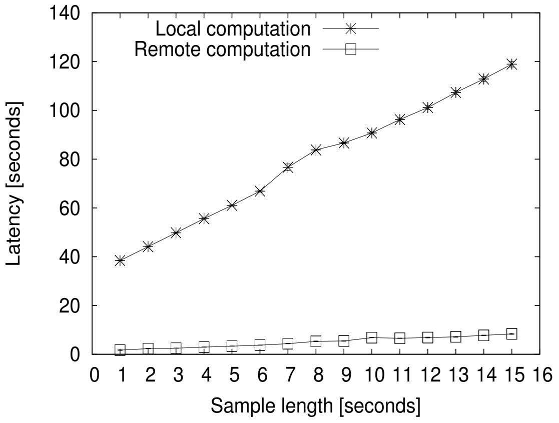


Figure 6.2: Speaker recognition latency vs audio sample length.

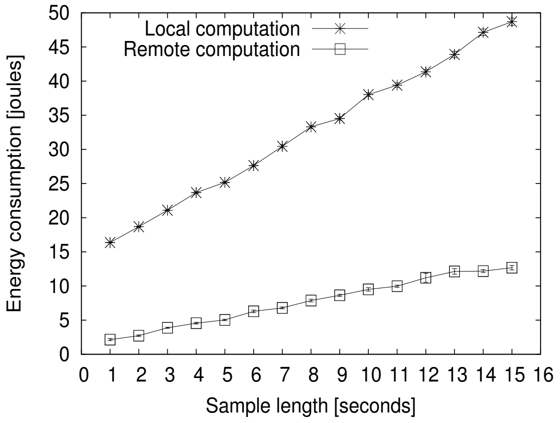


Figure 6.3: Speaker recognition energy consumption vs audio sample length.

computation, the process of comparing a voice sample with all the pre-loaded models is performed on a Nokia 6210 Navigator mobile phone that is equipped with an ARM 11 369MHz processor. In the remote computation case, an audio sample to be classified is transmitted over the 3G network using the *HTTP Connection* module of PyS60 to a powerful back-end server (Intel Xeon Octa-core E5506 2.13GHz processor, and 12 GB RAM). We used a total of 150 samples for this test (same as that used in the speaker accuracy test), each of length 5 seconds. A 5 second audio sample has a size of about 78KB. The energy consumption shown in the results in this subsection is end-to-end consumption including all computation and radio transmission costs. We measured energy consumption using the Nokia Energy Profiler.

The results of this test are shown in Figures 6.2 and 6.3. We can observe that the remote computation is more efficient both in terms of latency and energy consumption. For example, for a 5 second audio sample, the remote processing consumes 5 times less energy and 18 times less latency compared to that of the local phone processing. Further, as the audio sample size increases, the magnitude of the difference increases linearly, for example, for a 15 second sample, the latency difference between the local and remote processing is about 110 seconds whereas for a 5 second sample it is 57 seconds. In Chapter 5, Section 5.4 (Figures 5.4, 5.5 and 5.6), we have shown that, for the energy consumption, latency, and data sent over the network values corresponding to a 5 second audio sample, the computation offloading scheme selects remote computation given equal distribution of weights to all the dimensions as remote processing is more efficient than the local processing with respect to two out of three dimensions.

As discussed, offloading computation reduces energy and latency of intensive classification tasks, however, data needs to be transmitted to the cloud for remote processing, therefore, it uses the user's mobile data plan or Wi-Fi if it is available. Considering the rapid increase in the availability of number of open Wi-Fi networks, in future, the issue of using the user's data plan might subside. However, currently, not all phones have Wi-Fi. In these cases, local computation can be used, although, it leads to increase in the energy consumption while reducing the data sent over the network.

The Impact of Noise

We also conducted a test to evaluate the effect of noise on speaker recognition accuracy. We used Audacity [AUC], an open source cross-platform sound editor, to inject noise into voice samples. Audacity provides an easy way to add a particular type of noise into voice samples.

The real environmental noise is generated from many different types of sources such as machines or user activities, for example, in office spaces, it includes noise generated from moving desks, closing/opening doors, and operating coffee machines. In an environment such as the user's home, it might be generated from heating/cooling systems, vacuum cleaners, refrigerators or user activities such as cooking, cleaning, washing etc. Each of these sounds have different characteristics, further, the combinations of these noises produce even more types of noise. We, therefore, choose to use a random noise as it contains the characteristics of each of these sounds at some point or the other, moreover, its random nature makes it hard for the speaker recognition technique to account for the noise in its design. Accordingly, we selected Brownian noise (or random walk noise), as the change, or increment/decrement, from one moment to the next in this type of noise is random. We injected Brownian noise into all the test samples for their entire length with amplitudes ranging from 0 to 0.1 (% of amplitude), in increments of 0.02. Figure 6.4 shows the effect of Brownian noise on speaker recognition accuracy. As expected, the accuracy decreases as the amplitude of noise increases. The rate at which the accuracy drops is higher for the noise amplitude values up to 0.04 after which the rate of drop decreases, relatively.

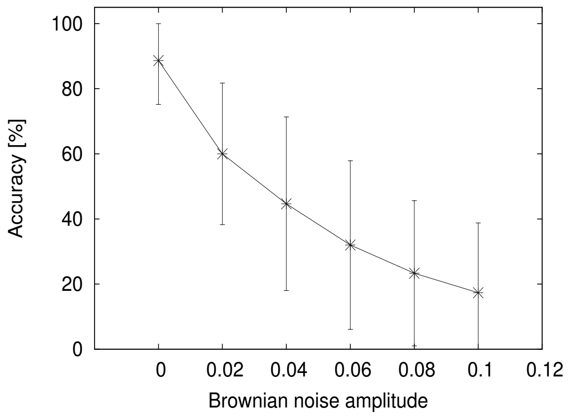
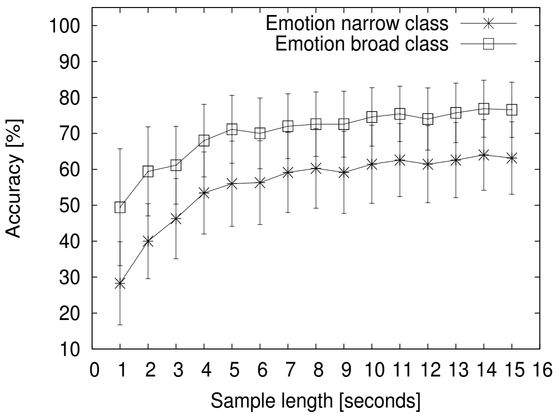


Figure 6.4: Effect of Brownian noise on speaker recognition accuracy (5 sec sample).



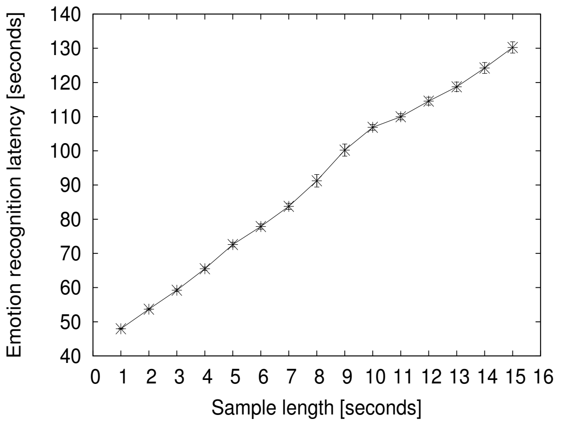


Figure 6.5: Emotion recognition accuracy vs audio sample length.

Figure 6.6: Emotion recognition latency vs audio sample length.

Emotion Recognition

In order to benchmark the emotion recognition subsystem, we used both test and training data from the Emotional Prosody Speech and Transcripts library [LDG$^+$02]. The advantage of using this library is that it is difficult for non professionals to deliver emotional utterances. An alternative is to use "natural" speech recordings (i.e., taken from everyday situations without acting). However, it is difficult to determine appropriate reference labels, required to evaluate performance on this speech, since many natural utterances are emotionally ambiguous. The use of a pre-existing library also allowed us to avoid explicitly annotating collected data with emotional labels. We used a total of 14 narrow emotions, which were then grouped into 5 broad emotion categories. For each narrow emotion, we used a total of 25 test samples per narrow emotion per sample length, resulting in a total of 350 test samples per sample length.

Figures 6.5, 6.6, and 6.7 show the emotion recognition accuracy, latency, and energy consumption, respectively, with respect to the audio sample length. As the sample length

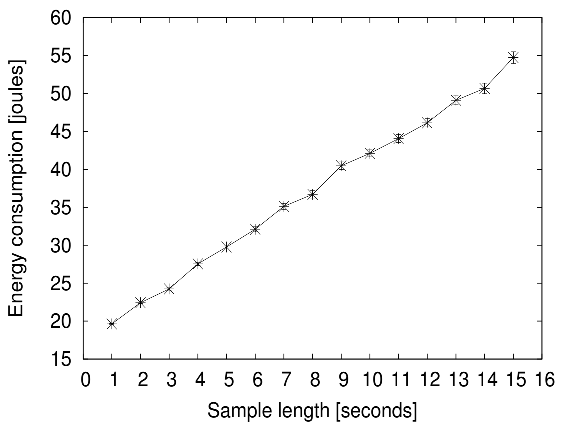


Figure 6.7: Emotion recognition energy consumption vs audio sample length.

Emotion [%]	Happy	Sad	Fear	Anger	Neutral
Happy	58.67	4	0	8	29.33
Sad	4	60	0	8	28
Fear	8	4	60	8	20
Anger	6.66	2.66	9.34	64	17.33
Neutral	6	5.33	0	4	84.66

Table 6.2: Confusion matrix for broad emotions.

increases, the accuracy improves, converging to about 71% for the broad emotions for sample lengths greater than 5 seconds. Based on the speaker and emotion recognition accuracy results (Figures 6.1 and 6.5), we used a sample length of 5 seconds in the EmotionSense system that is the point at which the convergence becomes evident. The confusion matrix for broad emotions for a sample length of 5 seconds is shown in Table 6.2. Among non-neutral emotions, anger has the highest accuracy of all. This is confirmed in [BOH83], where the authors show that intense emotions (like anger) are easier to detect than emotional valence. They also mention that emotions that are similar in intensity, like anger and fear (panic), are hard to distinguish: the same can be observed in our confusion matrix.

We also note that distinguishing between some narrow emotions in a group is difficult given the similarity of the utterances corresponding to them: for a sample length of 5 seconds, the narrow emotion "happy" matches "interest" with a probability of 0.28. Grouping these increases the accuracy of classification, which can be observed from Figure 6.5. The use of a limited number of broader emotional classes is also advocated by social psychologists [FBR98]: in general, classifying one's own emotions using narrow categories when filling self-report questionnaires is also difficult.

6.2.5 Social Study

After evaluating the accuracy of the system by means of the micro-benchmark tests, we conducted a social study. The main goal of the study is to show some examples of the type of data that can be collected through smartphones and the correlation analysis that could be performed on the collected data. We show this by analysing the emotion and speech patterns inferred by the EmotionSense system and studying the factors influencing these patterns by correlating these with sensor data. The data extracted by means of the EmotionSense system running on the mobile phones was also compared with the information provided by participants by means of traditional questionnaires.

Overview of the Study

The study was conducted over a period of 10 days and involved 18 participants. Each user carried a Nokia 6210 Navigator mobile phone for the total duration of the study. Users filled in a daily diary questionnaire for each day of the study which was designed by a social psychologist involved in our project following the methodology described in [BDR03]. We divided a day into 30-minute slots, and asked the users to complete a questionnaire on the activity/event they were involved in at a particular time of day. We also asked them if the event happened indoors or outdoors, their location, and if there were other people present (in particular, participants involved in our study). We also asked them to specify their mood at that time.

Participants

We searched for volunteers using snowball sampling [Goo61]. It is a technique where existing participants help in recruiting future participants from their acquaintances. Therefore, the growth of study group resembles that of a rolling snowball. We recruited 18 users from the Network and Operating Systems (NetOS) group, Computer Laboratory, University of Cambridge. The participants included 2 female and 16 male users and were Ph.D. students, researchers, and faculty members of the laboratory. Since the system did not require any user-phone interaction, the fact that the participants were technology-savvy is not a determinant factor in the outcomes of the experiment. Before the start of the experiment, we provided an overview of the application and the type of data it collects to the participants. We also explained them that they could opt-out of the study and request for deletion of their data at anytime during the experiment. We provided each of them with an unique identifier to complete online surveys.

Results and Discussion

We note that the results that will be presented in this subsection are some examples to show the type of data that could be collected from the phone's sensors such as accelerometer, Bluetooth, and microphone. One of our aims is to show using some examples that modern off-the-shelf smartphones provide an opportunity to understand the impact of various external factors on the user's behaviour as data can be collected from multiple sensor streams. Since these results are intended as examples, we have not performed any statistical tests, therefore, it is difficult to comment on the statistical significance of these results and these should only be viewed as examples.

We analysed the distribution of emotions detected, and also the effect of time of day, activity, and co-location, on the distribution of emotions. Figure 6.8 shows the distribution of "broad" emotions detected by the system during the experiment. We can observe that the application detected neutral emotions far more than other emotions. Figure 6.9 shows the distribution of emotions from the questionnaires completed by users, which shows a distribution of emotions that is only partly similar to that extracted by means of our system. From our discussions with the participants, we found that the users indicated the "happy" emotion as representing their mental state, but this might not necessarily mean that they expressed happiness verbally. This is one of the limitations of the system i.e., it can only infer the user's emotions exhibited verbally. We note that, even though we observed some correlation between the self-reported data and the automatic inferences by the application, this is just an example study to show the potential of automatic inference by smartphones and more detailed studies are required to confirm and validate these observations.

Modern smartphones with embedded sensors provide an opportunity to analyse the impact of various external factors by correlating the inferred emotional and speech patterns with sensor data.

Time of day. Figure 6.10 shows the distribution of emotions with respect to time of day. We can observe that the detection percentage of emotions is less in mornings than during afternoons and evenings. This is particularly true of most users in this experiment, who were conversing less in the mornings than afternoons or evenings.

Accelerometer. As discussed in Chapters 1 and 2, the accelerometer sensor in the phone can be used to infer the user's physical activity. Therefore, it enables automatic

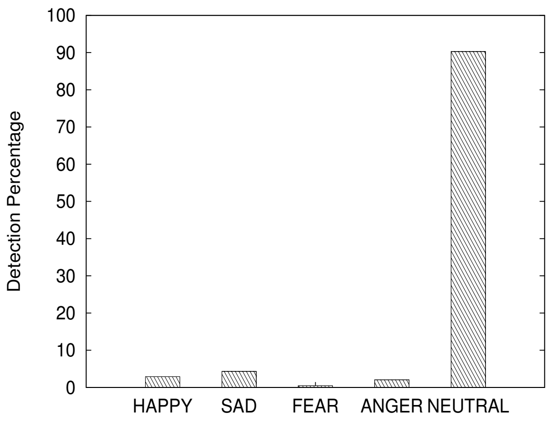


Figure 6.8: Distribution of broad emotions detected.

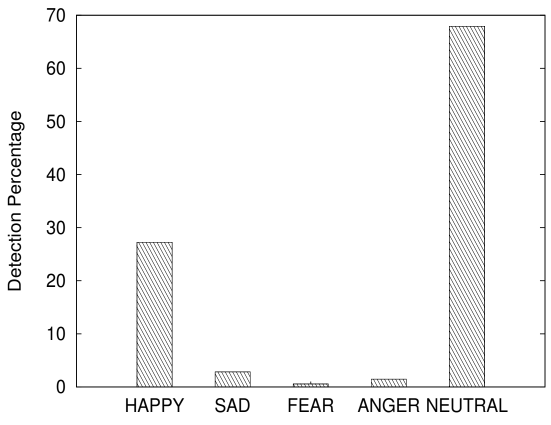


Figure 6.9: Distribution of broad emotions from daily dairies.

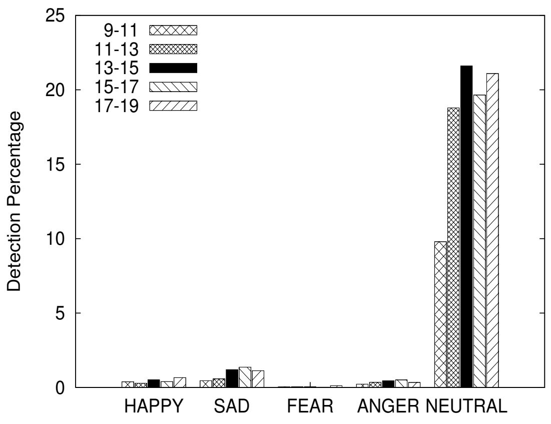


Figure 6.10: Distribution of broad emotions detected with respect to time of the day.

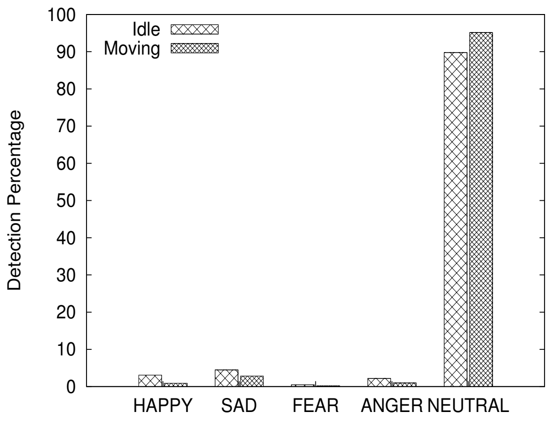


Figure 6.11: Distribution of broad emotions within a given physical state (idle/moving).

correlation analysis of the impact of the user's physical activity on her emotions. We studied the influence of the user's physical activity such as "stationary" or "moving" on her emotions. Instead of studying a global percentage of each of the emotions with respect to activity, we plotted the distribution of relative percentage of broad emotions when users are stationary and mobile. From Figure 6.11, we can observe that these distributions are very close, and the relative ordering is the same in both the cases.

Bluetooth. We used the Bluetooth discovery feature on the phone and a static mapping of Bluetooth addresses and participants' identifiers to infer the people co-located with the phone user. Figure 6.12 shows the impact of the number of co-located participants on the emotions of users. We can observe that the total number of emotions detected (for "neutral" category) in smaller groups is different from that in larger ones. This indicates that group size may impact the user's behaviour, and this is a factor that could be considered in social studies.

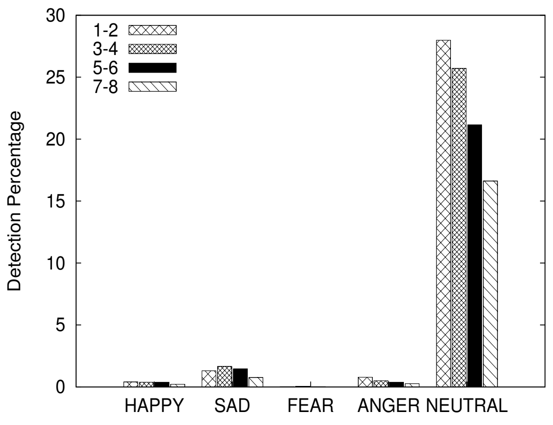


Figure 6.12: Distribution of broad emotions detected with respect to number of co-located participants.

Location. We were able to associate predominant non-neutral emotion with location categories (as labelled by the user): we found that the most common emotion in residential areas was "happy" (45%), whereas in the workplaces and city centre "sad" was the most frequently detected (54% and 49%, respectively). These observations show the potential of a deeper analysis of correlation between emotions and location, which can be investigated further from the socio-psychological perspective with more focused studies. We note that these observations might be specific to the cultural context of this study.

Microphone. EmotionSense can also be used to analyse the speech patterns of users with respect to time. Figure 6.13 shows the speech patterns of users over time, which reveal a considerable degree of consistency in most participants' verbal behaviour across a 6-day period (except for the first day of "user2" that seems unique to him/her). In addition to emotion and speech recognition, the data from the microphone sensor in the phone can also be used to understand the conversation leaders in meetings. We analysed the data collected from a meeting held during the experiment when 11 of the participants sat together and talked for 30 minutes. We identified *conversation leaders* in each time slot of length 5 minutes. The analysis is shown in Figure 6.14. We considered only the top five most active speakers for this plot. We used an audio sample length of 3 seconds for this analysis as the difference between the speaker recognition accuracies for sample lengths of 3 and 5 is only 3% (Figure 6.1). We can observe that "user4" is the leader in almost all the slots, except in 2nd and 5th slots, where he/she was challenged by "user2" and "user5", respectively.

We note that this study is intended as an example to show the potential of smartphones in collecting behavioural data about participants. The aim of the results presented from this study is mainly to show the type of data that can be collected using modern off-the-shelf smartphones and to also show using an example the correlation analysis that can be performed to identify the factors influencing the user's behaviour. We have not

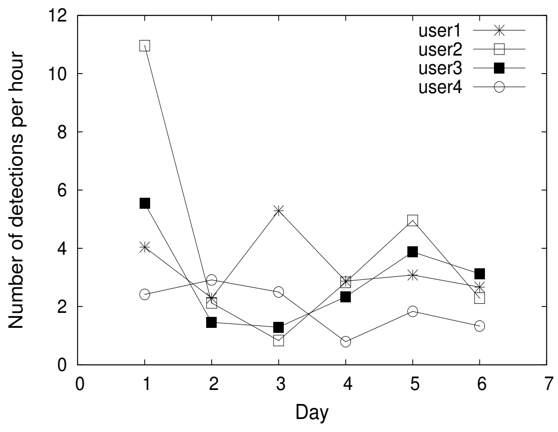


Figure 6.13: Variation of speech patterns over time.

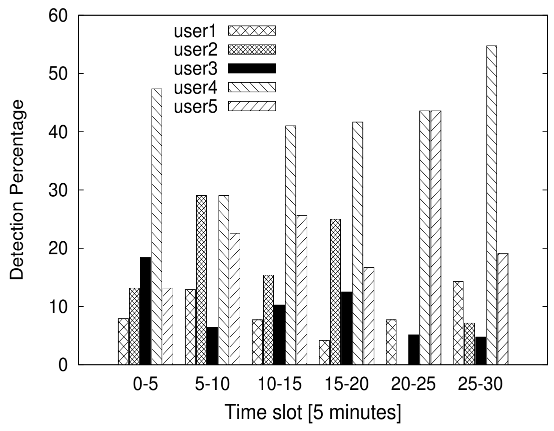


Figure 6.14: Speech time as percentage of the total speech time for all users.

performed any statistical tests as the results are intended as examples, therefore, it is difficult to comment on the statistical significance of the results. More focussed and large scale social studies are required in order to further confirm and validate the observations presented in this subsection.

6.3 WorkSense

In the previous section we presented a passive monitoring application for capturing the emotion and interaction patterns of the participants. In this section we show that smartphones can be useful for capturing the behavioural and collaboration patterns of users at the workplace and they can autonomously find the groups and interaction patterns of workers. We show this through the implementation and deployment of *WorkSense*, another example social application. We also show using this application that offloading sensing to the environment achieves considerable energy savings compared to the case of pure local phone sensing.

Although so far social sensing applications have remained primarily within the realms of entertainment and gaming, achieving reliable and accurate social sensing on mobile devices could allow the introduction of such systems as valuable business instruments with the potential to improve work performance and team cohesion. In business environments work performance is typically evaluated and monitored according to certain work-related metrics: whether deadlines are met, tasks are progressing according to schedule, etc. However, these metrics may be very limited and there is a vast amount of information that is currently not captured by any available technology: these include the daily interactions of people at the workplace, and the effect of these interactions on their work. Chats between colleagues in the corridor and lengthy discussions during a coffee break are examples of behaviour that, if recorded, could help identify “key” players in the resolution of issues, or even identify how teams could improve collaboration. Studies [LWA+08, OP10a] have shown that meetings and informal interactions at the workplace can have a positive impact on the work of teams and individuals. This kind of study has been conducted with specialised devices (often RFID tags) which must be carried by participants throughout the study.

In this section we present WorkSense, an example social sensing application that utilises the sensing offloading scheme presented in Chapter 4 to achieve accurate and energy efficient sensing of social activities at the workplace. WorkSense uses the sensing offloading scheme as it mainly targets office environments, which might be instrumented with sensors. The primary aim of the WorkSense application is to automatically infer meetings and collaborations among users at the workplace. We deployed the WorkSense application in our research institution for over a month. The application is deployed as part of the system presented in Chapter 4. The application uses data from the mobile phone sensors (accelerometer, Bluetooth, microphone), and opportunistic interaction with environmental sensors that have been deployed in the office environment (Bluetooth scanners, desk sensors, microphones) using the sensing offloading scheme. Based on this data, the application can detect various interactions and activities, and automatically infers collaborations and meetings of users. Furthermore, by combining sensed information with work-related data (calendars, application activity loggers), it can help in understanding the influence that certain social interactions may have on their work.

We evaluated the WorkSense application through a real deployment with 11 users for a month in a working environment. We show that it is able to infer the effect of various interaction and social patterns on the work of the users by capturing data from mobile, infrastructure sensors, and combining them with work-related data about the users. We also studied the performance benefits of offloading sensing to the environment using the scheme presented in Chapter 4.

6.3.1 Workplace Monitoring

A number of social studies have attempted to evaluate the influence of social behaviour at the workplace [LWA+08, OP10a, WOKP10]. They showed that social interactions at the workplace can have positive impact on work performance, team cohesion, and in general on employees’ well-being. However, most of these studies mainly use surveys and self-reports or purpose-built devices [CP03, MP01] that workers are asked to wear during the study.

Our aim in this section is to build an example social sensing system in which social interactions are automatically detected during the day and associated with the work activities for prolonged periods, where a worker is able to identify the person who is most influential in dealing with particular tasks, or social activity patterns that have either a positive or negative influence on their work. In addition to self-reflection, such a system could allow workers to compare their social behaviour pattern with others. Olguin and Pentland [OP10b] have identified worker types according to social behaviour. Awareness can influence behaviour and possibly lead to behavioural change. Apart from the social impact of such tools, practical use can include the association of people with particular activities, based on their social interaction patterns. In capturing social interactions at the workplace, the use of mobile sensing technologies, and in particular mobile phone sensing, can be invaluable. However, energy limitations on mobile phones can hamper usability and, inevitably, acceptability of such systems by end users. Therefore, we use the system architecture presented in Chapter 4 to exploit the sensors available in the infrastructure to perform efficient social sensing.

6.3.2 Application

The main goal of the WorkSense application is to detect interactions and collaborations at the workplace. The design of the WorkSense application is motivated by a number of studies that have shown the importance of understanding organisational behaviour. For example, in [LWA+08] 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 organisation.

We exploited a number of sensing modalities that were available in our office environment using the sensing offloading framework presented in Chapter 4 to design *WorkSense*: an application that aims to infer collaboration and meetings of users, and can be useful to understand the impact social interactions may have on their work. Our target is to use mobile phones to monitor and visualise work behaviour and opportunistically use building sensing infrastructure to minimise the energy cost while not compromising on accuracy. The WorkSense application consists of a mobile phone application that is able to detect social interactions, co-location patterns, and a back-end service used for data aggregation and sharing. We use the system presented in the Chapter 4 to deploy the WorkSense application.

More specifically, WorkSense includes the following functional components:

Mobile phone application. The mobile phone application is implemented on the Android platform and utilises the sensing offloading service. The application captures the mobility patterns of users during working hours (visited offices, meeting rooms, common rooms, etc.) and conversations with colleagues. This information is used to infer social context, mining working patterns for a given time of day, and to offer awareness to the user [LWA+08]. WorkSense uses the conversation detection classifier of the EmotionSense system to detect conversation.

Meeting detection. A meeting is considered to take place when two or more users are co-located for more than a pre-set time threshold (currently 15 minutes) and are in a conversation. In the definition, we do not specify as a prerequisite that the location should be a meeting room, as we wanted to capture informal meetings that sometimes 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 show changes in the user's behaviour before and after a particular meeting. Such information can allow the user to identify meetings that have significant influence on their work, or people who may have an effect on their performance.

Collaboration detection. Although people in our research institution typically form groups that may collaborate within the context of a particular research project, the WorkSense application attempts to capture a more objective picture of how collaboration takes place in the research lab. Sometimes a user who is not officially assigned to a project may play an important role. Collaboration detection is achieved by applying a community detection algorithm to the co-location data. However, as there are cases where two or more colleagues may share an office, a straight-forward application of the community detection algorithm would result in communities that are heavily biased towards people sharing offices, even if they do not typically collaborate. In order to overcome this problem, we defined an *intended visit* as the case when a person visits another person with the intention of conversing. We define the intended visit as: $deskEmpty(u_1) \wedge colocated(u_1, u_2) \wedge conversation(u_1)$. 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. The Collaboration Detection applies a community detection algorithm over the directed graph of conversations between users. The dataset consists of a sliding window of data applied over the last two weeks. The system uses the Louvain community detection algorithm [BGLL08] to detect groups and collaborations.

Work activity monitoring component. The work activity monitoring component is implemented in Java and is supported for Linux and Mac OS X platforms. The component periodically logs details about all the applications running on the user's computer, and also detects the current active application. The details about all the running applications are obtained by logging the output of the `top` command that gives the list of processes running on the user's computer. The module used to detect the active window on the Mac OS X platform is implemented in AppleScript[2], and on the Linux platform, we used `xwininfo`[3] command that is a utility for querying various details about the windows in the Linux platform.

In addition, the application also needs an indoor localisation component to detect the location of the user. The indoor localisation was performed using Bluetooth devices

[2]http://developer.apple.com/applescript

[3]http://www.xfree86.org/4.2.0/xwininfo.1.html

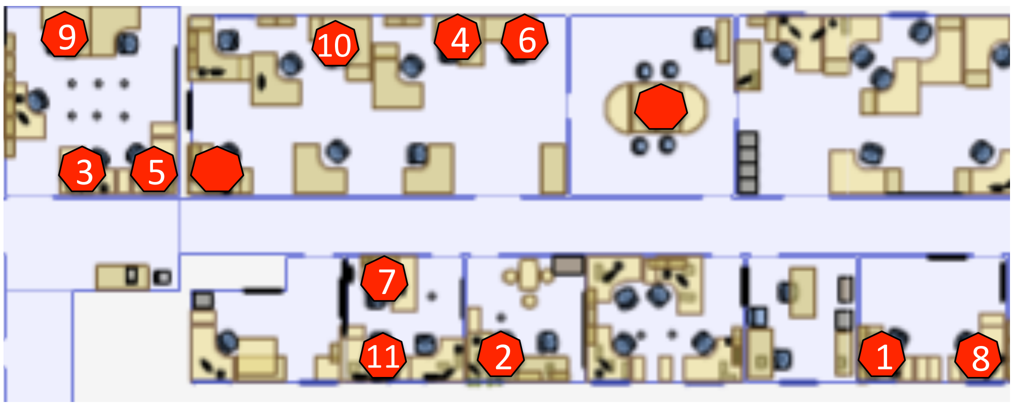


Figure 6.15: Sensor nodes were installed on 13 desks including two relay nodes.

in the environment and using a static mapping of devices and locations. Location was detected by comparing the Received Signal Strength Indication (RSSI) values of the co-located Bluetooth devices, and the device with the highest RSSI value is mapped as the location of the user. From a design perspective, WorkSense is a mobile phone application and its operation *does not require* the presence of sensors embedded in the environment; however, when such infrastructure is available, its energy performance can be dramatically improved and the accuracy of the gathered information can be increased as well.

6.3.3 Deployment

We deployed the WorkSense system in a working environment for over a month. The main aim of the deployment was to show through an example application that it is possible to analyse and infer work activity habits and patterns, interaction and activities, and collaboration patterns at the workplace using the data collected from the sensors in smartphones and the infrastructure. We recruited 11 participants from the Computer Laboratory, University of Cambridge, United Kingdom, using the same procedure as described in Section 6.2.3. The participants are students and staff of the University. During this period each user carried an Android phone (Samsung Galaxy S or HTC Desire). The mobile system was able to utilise existing sensing infrastructure that was deployed in our institution (Figure 6.15) using the sensing offload scheme. This included desk occupancy sensors (imote2 sensors), Bluetooth sensors, and conversation detection infrastructure (see Chapter 4, Section 4.5).

The WorkSense back-end component is implemented in PHP and Python and is deployed on a powerful back-end server (Intel Xeon Octa-core E5506 2.13GHz processor, and 12 GB RAM). The web API interface to the mobile and desktop components were provided through PHP scripts. A Python-based cloud module is implemented to correlate the data streams from social sensing (recorded by the mobile applications) and work activities (captured by the task-loggers running on desktop/laptop machines), based on the timestamps.

Sensing Offloading

As the WorkSense application is aimed at monitoring users at the workplace, this deployment provided us an opportunity to evaluate the benefits of the sensing offloading scheme. In this subsection, we present the results of the battery performance of the sensing offloading scheme presented in Chapter 4.

The aim of this evaluation is to show the advantage of exploiting sensing infrastructure over the pure local phone sensing on real smartphones. The phone was running the WorkSense application, which uses two social sensing classifiers described in the previous subsection to automatically detect interactions and collaboration patterns: (i) a speaker identification classifier based on the microphone sensor and (ii) a co-location detection classifier based on the Bluetooth sensor. To compare the difference in battery life of the phone *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 (local 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 on 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 measured the battery level using the *BatteryManager* API [4] of the Android platform.

Most operating systems provide an estimate of the remaining battery based on power models, therefore, the remaining battery charge value provided by the operating system is subject to errors due to minor differences between the phones such as temperature and recharge cycle count (or how much the battery has been used). To minimise this error, we tested the schemes on two different phones but of the same model and make, and then calculated the average of the results. Further, since the aim of this test is to measure the relative differences among the schemes, this evaluation provides us a way to estimate the relative gains of the sensing offloading scheme over the other schemes. We note that the absolute numbers provided in this subsection for the battery lifetime have only been used for relative comparison and these might vary based on the test setup.

We 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. The average consumption over multiple runs is shown in Figure 6.16. The results show that, when using the threshold based offloading scheme and exploiting the sensing infrastructure, the battery lasted up to 45% longer compared to the case of local phone sensing with Wi-Fi on, and 35% longer when compared to the

[4]http://developer.android.com/reference/android/os/BatteryManager.html

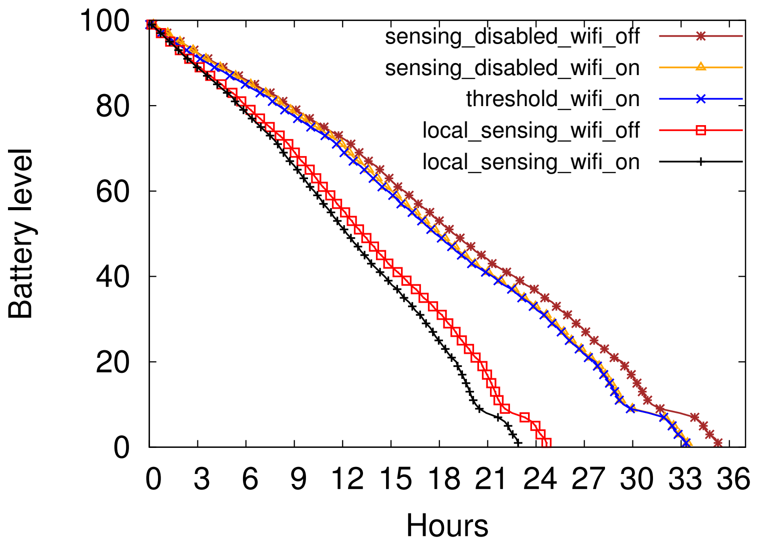


Figure 6.16: Battery drain of Samsung Galaxy S phone for various scenarios.

case with the Wi-Fi off. Furthermore, the energy cost of using the system 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 support for the long-term deployment of sensing applications, by significantly minimising the impact on the phone's battery life.

Social Results

In this subsection, we present the results related to the social and behavioural patterns of the participants to show some examples of the inferences that could be drawn, for example about collaborations, using the data collected by smartphones.

Meeting detection analysis. One of the primary motives for the design of WorkSense was that social sensing has the potential to capture views over interaction and activities at the workplace. The most expected result of the WorkSense application was the detection of formal and informal meetings. The *Meeting Detection* service was able to detect the exact times at which meetings took place in the laboratory. Furthermore, by utilising the calendar information such meetings were populated with appropriate meta-data. Figure 6.17 shows a snapshot of a time-line where the calendar schedule is contrasted with the actual meetings as they were detected by WorkSense. In this snapshot we see how a person that has consecutive meetings can slightly adapt the schedule according to the duration of previous meetings. In this case many meetings were moved forward: this was not something updated in the calendar. A similar observation was reported in [LOIP10], where the authors showed that the calendar does not accurately represent reality as genuine events are hidden by placeholders and reminders. Furthermore, WorkSense was able to capture a number of unscheduled informal meetings typically taking place at the work-

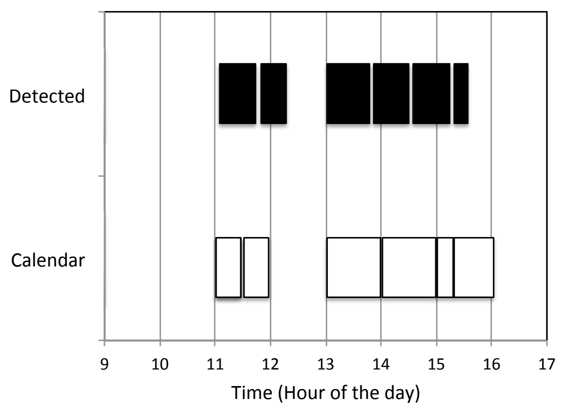


Figure 6.17: Timeline of detected v.s. calendar meetings.

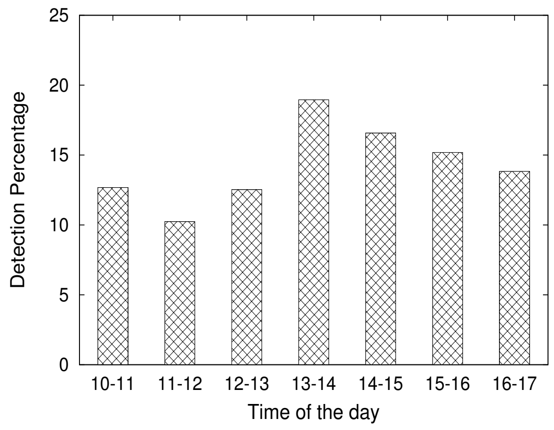


Figure 6.18: Conversation patterns with respect to time of the day.

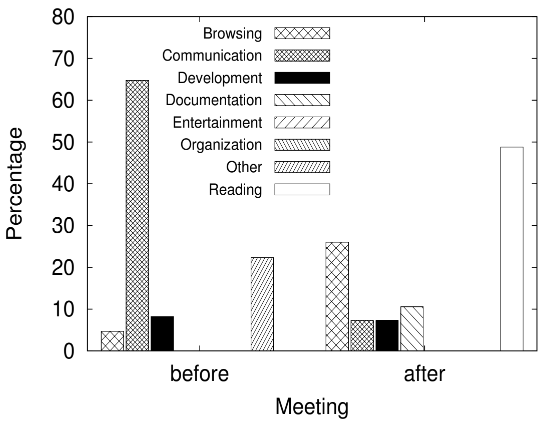


Figure 6.19: Work pattern of User 7 before and after a specific meeting.

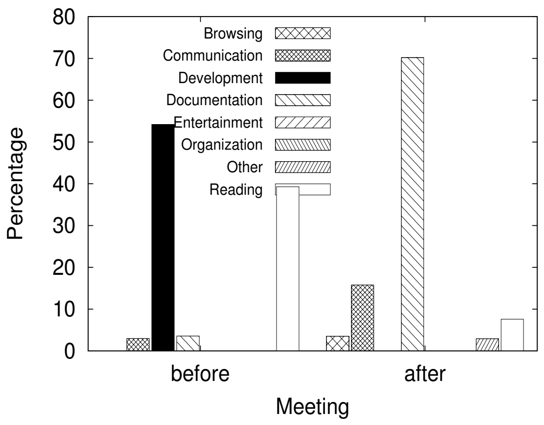


Figure 6.20: Work pattern of User 7 before and after a different meeting.

place. Although the impact of such meetings may not be immediately obvious, providing information about such meetings may prove to be a valuable tool.

Activity type and user specific analysis. Information on which activities mostly take place at specific times of the day can be detected: an example is shown in Figure 6.18 where the detected conversations are shown. Furthermore, interesting results are obtained by comparing the work activities of a user before and after particular meetings. To capture the work activities that act as ground-truth to show the varying level of effect of different meetings, we installed task-loggers on the users' computers as part of the work activity monitoring component described in the previous subsection. The task-loggers periodically log details of all the applications running on the user's computer and also detect the current active application. The data on meetings detected by the WorkSense can be fused with data from the task-loggers in the back-end server to extract interesting results: In Figures 6.19 and 6.20, it can be seen that the type of activity of User 7 changes drastically after two different meetings.

Community detection analysis. Before the deployment of the WorkSense application, we collected the *ground truth* of how the participants are organised into work teams. Users were asked to offer details about the people they currently collaborate with, the teams they work in and the projects they work on. Figure 6.21 shows the different groups and project teams as reported by the users. The users were divided into two research groups. During the experiment three major projects were taking place. Each user belonged to a single group and worked on a project.

We used WorkSense to automatically detect such communities. To do this, we first create a weighted graph of detected collaboration where a link between two users represents the length of time they spent in meetings and discussion. Afterwards, we apply the *Louvain* community detection algorithm [BGLL08] to detect groups and projects that users belonged to. The Louvain method is a simple and efficient method for identifying communities in networks and is one of the most widely used methods. It is a hierarchical greedy algorithm and operates in two phases, repetitively: first, it searches for small communities by considering nodes one by one, and then it combines nodes of the same community and forms a new network. The method uncovers hierarchies of communities and allows further identification of sub-communities, sub-sub-communities and so on. The Louvain's method is also shown to be computationally efficient: the method runs in time $O(nlogn)$ with most of the complexity incurred by optimisation at the first level.

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 thickness of edges in the graph represent the weight/strength of the link, and the colour of a node represents its community. Figure 6.22 shows the Level 2 communities where WorkSense was able to identify the separation between people belonging to different departmental groups. We can observe that the algorithm is able to group users into two communities:
Group 1: U.2, U.3, U.4, U.6, U.7, U.9, U.10, U.11
Group 2: U.1, U.5, and U.8.

The groups are in accordance with those reported by the participants (Figure 6.21). The spatial placement of nodes is generated using a spring model (Fruchterman-Reingold force-directed algorithm [FR91]): the larger the spatial difference between two nodes, the lesser the collaboration/communication between them. More interestingly though, the output of community detection when operating at Level 1 (Figure 6.23) 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 had been working on, even within the same group. The communities detected are in agreement with the project groups reported by the users (Figures 6.21 and 6.23). Furthermore, the WorkSense application was able to identify cross team collaboration and pinpoint people acting as *bridges* extending collaboration links across teams.

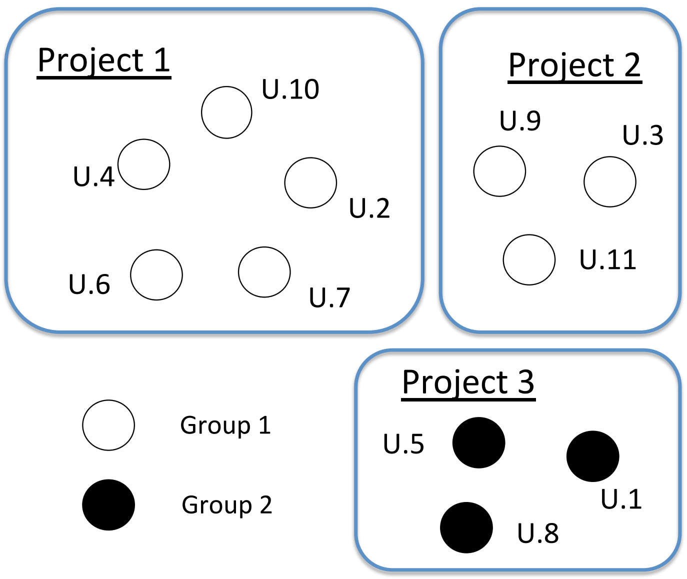
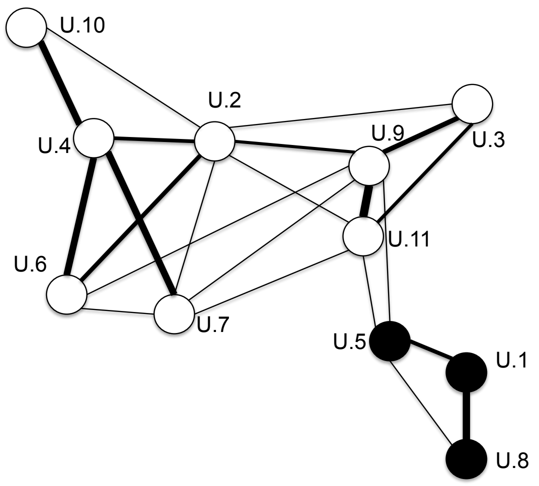


Figure 6.21: Communities as reported by the users.



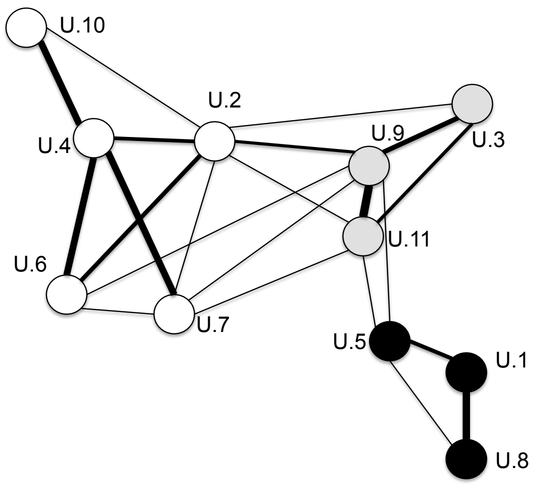


Figure 6.22: Automatic detection of groups (level 2 communities).

Figure 6.23: Automatic detection of projects (level 1 communities).

The analysis and results presented in this subsection serve as another example to show the potential of smartphones for conducting social studies. We note that these are just few examples and initial steps to show the potential of smartphone sensing and more studies are required to precisely gauge the accuracy and validity of these social and behavioural inferences.

6.4 SociableSense

In the previous section we presented an application for monitoring the behaviour of users at the workplace. Smartphones are not just useful for passive monitoring, but also in *closing the loop* by providing feedback and useful statistics about the user's activities. As another example, in this section, we present an application that estimates the sociability of users and provides *gaming* features to encourage users to become more sociable.

In this section, we present the design, implementation, and deployment of *SociableSense*, a smartphones based social sensing application intended to provide real-time feedback to users in order to encourage them to foster their interactions and improve their relations with colleagues. The core of the application is a *social feedback component* that estimates the sociability of users (i.e., a quantitative measure of the quality of their relations) based on interaction and co-location patterns extracted from the sensed data at run-time and provides them with feedback about their sociability, the strength of relations with colleagues, and also alerts them to opportunities to interact. SociableSense utilises the services of the adaptive sensing scheme presented in Chapter 3 to efficiently sample the sensors and the computation offloading scheme presented in Chapter 5 to efficiently compute the classification tasks.

To demonstrate how such an application can be useful in practice, we conducted a social study in an office environment where 10 participants carried mobile phones for two working weeks. More specifically, in this section, we present the design, implementation, and evaluation of a system for quantitatively estimating the sociability and relations of users in the office environments. 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*.

6.4.1 Relations in Workplaces

Social science researchers have been devoting their attention to investigating behaviour and interaction of users in the workplaces for a long time, trying to answer questions about different aspects of user behaviour. *What are the interaction and colocation patterns among users in the office or corporate environments? Do people socialise more in personal office spaces or in common spaces like coffee rooms? Which work-group members socialise with one another and why?* Considering evidence that work-groups with highly connected team members outperform less connected ones [LWA$^+$08, OP10a, WOKP10], the importance of these research questions is clear: Understanding how work-groups socialise naturally and identifying the factors that moderate socialisation 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 socialise 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 time of the day or day of the week?*

Investigations of workplace socialising tend to rely primarily on self-report methods. Although such methods can be useful and effective for assessing attitudes and beliefs, considerable evidence suggests that reports of behaviour tend to be inaccurate [PR91]. Laboratory-based studies enable researchers to observe behaviour instead of relying on behavioural 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.

Smartphones can prove invaluable for continuous monitoring of people in their environment as shown with the previous example applications, EmotionSense and WorkSense. Since almost everyone has a mobile phone, they are a perfect platform for conducting behavioural monitoring studies. In this section, we show how smartphones can be used to provide realtime feedback to users, using an example application that estimates the sociability of users and alerts them to opportunities to interact.

6.4.2 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 relations with colleagues. We measure the strength of a user's relations and his/her overall sociability based on *network constraint* [Bur95][5]. 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, a 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:

$$N_i = \sum_j (p_{ij} + \sum_q p_{iq}p_{qj})^2, q \neq i, j; j \neq i \tag{6.1}$$

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 co-location and interaction patterns. We define colocation of a pair of users as being in proximity to each

[5]An alternative and in a way similar measure is the *kith index* [WOKP10].

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 and the speaker identification classifier described in the EmotionSense application (Section 6.2)[6]. We refer to the network constraint for colocation network as *colocation network constraint* (NC). This is calculated as follows:

$$NC_i = \sum_j (pc_{ij} + \sum_q pc_{iq}pc_{qj})^2, q \neq i, j; j \neq i \tag{6.2}$$

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), which is calculated as follows:

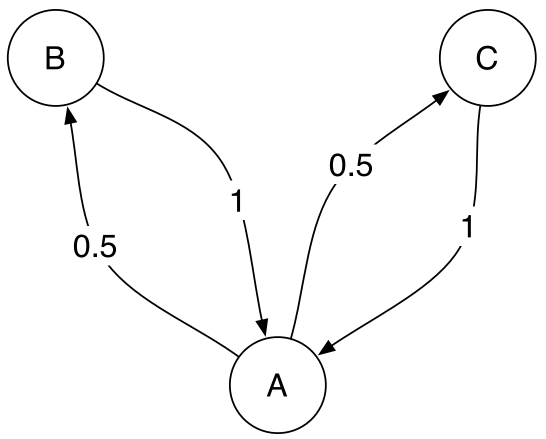
$$NI_i = \sum_j (ps_{ij} + \sum_q ps_{iq}ps_{qj})^2, q \neq i, j; j \neq i \tag{6.3}$$

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 the user has spent time interacting with many colleagues, i.e., he/she is more sociable.

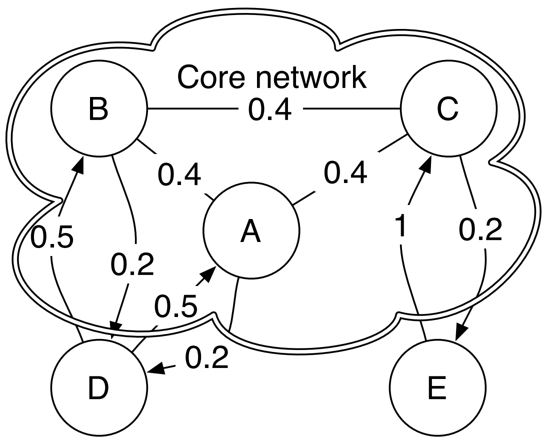
In a social network of n nodes, the strength of relations of a user i with respect to colocation is calculated based on pc_{ij} where $j = 1, 2, \dots n, j \neq i$ and with respect to interactions is calculated based on ps_{ij} where $j = 1, 2, \dots n, j \neq i$. We call the user i where $i = min\{NC_k, k = 1, 2, \dots n\}$ (i.e., the user with least colocation network constraint value) the *mayor* of the group with respect to colocation, and similarly the user j where $j = min\{NI_k,\ k = 1, 2, \dots n\}$ (i.e., the 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 socialise more by adding the *competition* and *gaming* factors.

We use two examples to demonstrate the quantification of sociability through the network constraint. Figure 6.24(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 and 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 the 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 between two contacts, he/she is less constrained than B and C, therefore A is more sociable. Figure 6.24(b) shows a social network of five nodes. Let us assume that there is a core social network consisting of nodes

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



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



(b) C is more sociable than A.

Figure 6.24: Sociability measurement examples.

A, B, and C, and two new persons D and E have recently joined the network. D has interacted with two people in the network, whereas E has connected only with one person. Therefore D is connected more strongly than E. The constraints 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 less 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 of users is of very high interest to social scientists and corporations, as it helps them to understand social patterns of users at a great level of detail. It can also be used to create a positive impact on productivity [LWA$^+$08, OP10a, WOKP10].

6.4.3 Indoor Localisation and Alerts

The indoor localisation feature in the SociableSense system is based on the Bluetooth sensor, and is implemented to identify the users in social locations to notify colleagues of opportunities to interact, 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 coarse grained localisation[7]. The main reason for using this method is that since the system is already required to scan for Bluetooth devices to identify co-location of users, it need not expend additional energy 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 smartphones (for example, Nokia 6210 Navigator), and GPS does not generally work in indoor office locations.

[7]We also note that more fine-grained solutions based on Wi-Fi fingerprinting and coarse-grained solutions like logical localisation [ACRC09] can be integrated into this module, but since our aim is to distinguish between work and social spaces, we limit this feature to the Bluetooth based technique.

Our main aim is to distinguish between work and social spaces. Work locations are those in which users work and spend most of their office time. Social or sociable locations are those where users socialise and spend time during breaks, like coffee rooms, common rooms, and game rooms. When a user is in a social location, an alert is sent to all other participants, so that interested people can join the user and socialise with him/her.

6.4.4 Social Study

We conducted a social study to show how smartphones can assist in understanding the impact of feedback mechanisms on users. The application was deployed as part of a system that uses the adaptive sensing scheme presented in Chapter 3 and the computation offloading scheme presented in Chapter 5.

Overview of the Study

We conducted the study for a duration of two working weeks (10 days) involving 10 users. The participants were recruited from the Computer Laboratory, University of Cambridge, using the same procedure as described in Section 6.2.3 of this chapter. Most participants of this study were also part of the previous studies described in Sections 6.2.3 and 6.3.3. In particular, seven participants of this study were also part of the previous studies. We divided the experiment into two phases each lasting 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 showed the following to the users: sociability, strength of their relations (inferred using the microphone and Bluetooth sensors), activity levels (inferred using the accelerometer sensor), and alerts about the users in sociable locations (using the Bluetooth and network connection such as 3G or Wi-Fi). During the study each user carried a Nokia 6210 Navigator mobile phone. We identified five main locations where users generally spend most of their time. We categorise 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 socialise or have breakfast/lunch. These two categories of location are in different parts of the building, i.e., Bluetooth ranges were non-overlapping, therefore, localisation error is negligible. The study only relied on the data from smartphones and no self-reported data was used. In particular, through this example study, we aim to show that it is potentially possible to observe the relative differences in the behaviour of users using data from the phone's sensors.

Results and Discussion

Through the deployment, we studied the effect of feedback mechanisms on the sociability of the users. Figures 6.25 and 6.26 show the *colocation network constraint* and *interaction*

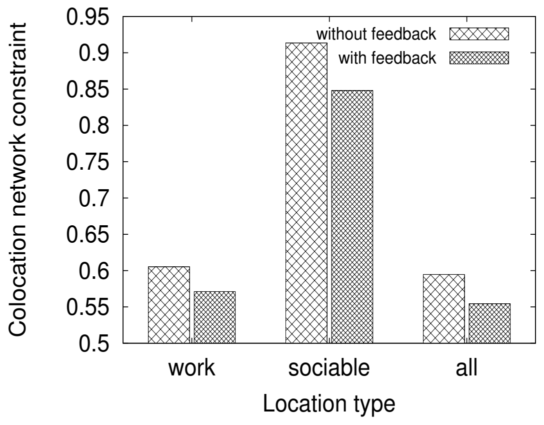


Figure 6.25: Colocation network constraint vs location types.

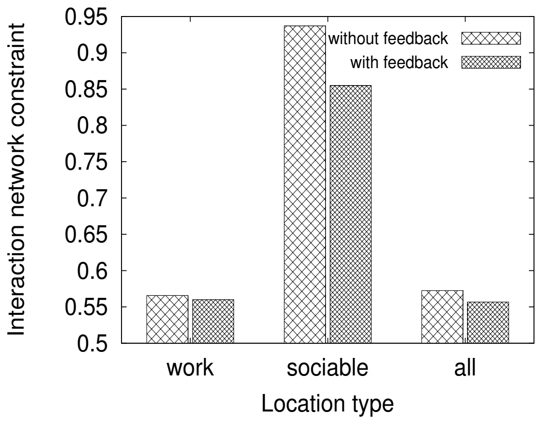


Figure 6.26: Interaction network constraint vs location types.

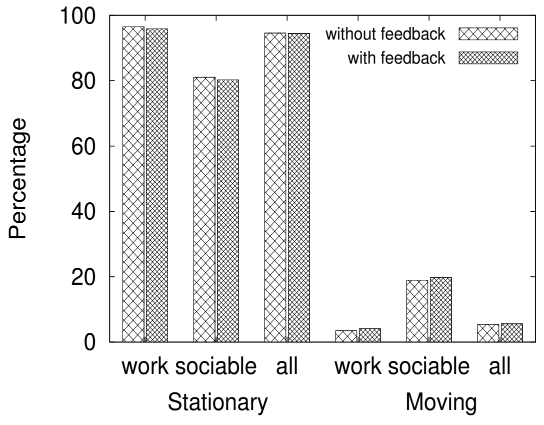


Figure 6.27: Activity levels of users in various location types.

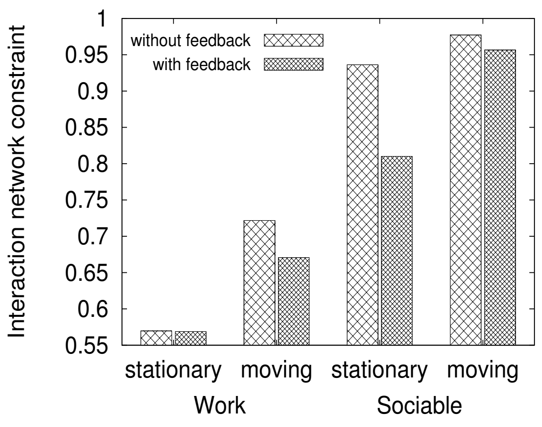


Figure 6.28: Network constraint vs activity levels.

network constraint, respectively, 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. However, we note that there might be many factors influencing the behaviour of users, therefore, the results presented in this subsection are just some examples of what can be potentially possible with smartphones and how they are helpful. Through this study, we wanted to show that the differences with and without feedback mechanisms can be analysed using data collected by mobile phones.

Accelerometer. We also analysed the effect of feedback mechanisms on the level of activity of the users using the phone's accelerometer sensor. As discussed in Chapters 1 and 2, the accelerometer sensor in the phone can be used to infer whether a person is moving or not. Figure 6.27 shows the level of activity of users in both the phases of the experiment, where we observe fairly consistent behaviour. A more deeper analysis can be performed by measuring the correlation between the level of activity and interaction network constraint, which is shown in Figure 6.28.

The social results presented in this section show that by conducting studies in different phases, the relative differences of the behaviour of users can be studied. We note that this is just an example study and more rigorous studies need to be conducted by social scientists to infer the precise impact of feedback and alert mechanisms.

Adaptive Sensing and Classifiers

The SociableSense application uses the adaptive sensing scheme for the following classifiers and sensors.

- **Conversation detection classifier (Microphone).** This is same as that described in Section 6.2.2 of this chapter and is based on the phone's microphone sensor. Since the interaction network constraint is estimated by inferring the speech patterns of users, "conversation" or "speaking" is considered as an interesting event and "silence" as an uninteresting event in the adaptive sensing scheme. The adaptive sensing scheme for this combination of interesting and uninteresting events has already been evaluated in Chapter 3, Section 3.5, where we have shown that the energy consumption is 43% less than that of the continuous sensing scheme for the same level of accuracy.
- **Activity recognition classifier (Accelerometer).** This classifier infers whether the user is moving or not using the accelerometer sensor and is same as that described in Section 3.5. For this classifier, "user moving" event is considered as an interesting event and "stationary" event as an uninteresting event. We have already presented the energy results for this combination of event classification in the same section. The results show that the energy consumption of the adaptive sensing scheme is 42% less than that of the continuous sensing scheme.
- **Indoor localisation and co-location detection (Bluetooth).** This classifier infers the indoor location of the user using the phone's Bluetooth sensor and fixed Bluetooth anchors deployed in the environment, and her co-location using the Bluetooth radios on the participants' phones. The change in the number of Bluetooth devices as sensed by the phone's Bluetooth sensor is considered as an interesting event otherwise as an uninteresting event, which is same as that considered in Section 3.5. The results presented show that the adaptive sensing scheme consumes 50% less energy than that of the continuous sensing scheme for this classifier.

Computation Offloading

The SociableSense application uses the computation distribution service described in Chapter 5 that computes the classification tasks either locally on the phone or remotely in the cloud considering the user's requirements. We have already shown the advantages of using the computation offloading in Chapter 5, Section 5.4, and further in this chapter when discussing about the results of the EmotionSense system (Section 6.2.3).

In this study we gave equal importance to all the dimensions: energy, latency, and data sent over the network. The combination of weights is the same as that of *S4* in Chapter 5, Section 5.4. Thus, the computation distribution scheme selected the configuration *C4* (speaker identification task computed remotely and other classification tasks computed locally) out of 16 possible combinations, as the selected combination is best with respect to two (energy and latency) out of three dimensions. The use of the computation distribution scheme led 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 in which all the classification tasks are computed remotely.

6.5 Related Work

In recent years, we have witnessed an increasing interest in the use of ubiquitous technologies for measuring and monitoring user behaviour [Pen08]. Experience sampling mechanisms [FCC$^+$07, MKL$^+$10] can be used to evaluate human-computer interaction especially for mobile systems since the use of the devices is not restricted to indoor environments unlike laboratory studies. The most interesting example of such systems is MyExperience [FCC$^+$07], a system for feedback collection triggered periodically, partly based on the state of on-board sensors. MyExperience does not include components for speaker, emotion recognition or automatic classification. Furthermore, the self-report based mechanisms are prone to memory errors [Tou99] and are also found to be biased towards positive experiences [PR91]. In [MKL$^+$10], the authors examined the use of a mobile phone based experience sampling application for cognitive behavioural therapy. The application collects data about the users' emotions and their scales. However, this data has to be manually entered into the system by the user. In contrast to these works, the proposed social sensing systems allow for automatic detection of behavioural aspects through emotions and speech patterns. Most of the existing systems are passive monitoring systems, unlike SociableSense, which can be used to measure and provide feedback to users.

The authors of [JHYGP09] model dominance in small group meetings from audio and visual cues, however, they do not model emotions. The MoodSense system [LLLZ11] can infer the user's mood from information already available in modern smartphones such as SMS, email, phone calls, application usage, browsing history, and location information.

The MIT Mood Meter [HHDP12] is a computer vision based system that automatically detects and counts smiles in a large community from data captured through cameras installed at various locations. For smile analysis, first they use a classifier to perform face detection and then they extract several geometric properties of the faces to predict the intensity of a smile. The approaches of these systems are orthogonal to our approach for detecting emotions where we use the microphone sensor in the phone. StressSense [LRC+12] is a smartphones based platform that uses the microphone sensor to detect stress in the user's voice. The system uses many acoustic features from the recorded audio for classification such as standard deviation of pitch, perturbation, and the rate of speaking.

Recent systems for quantitatively measuring aspects of human behaviour using purpose-built devices include the Sociometer [CP03] and the Mobile Sensing Platform [WBCK08]. A sociometer (or sociometric badge) is a wearable electronic device that can automatically measure face-to-face interaction, conversational time, physical proximity to other people, and physical activity levels using the sensors embedded in the device, similar to the phone. The Electronically Activated Recorder (EAR) [MP01] records audio samples in the user's environment using an audio recording device for 30 seconds once every 12 minutes. The audio files are coded offline by social scientists by manually listening to them.

Many studies have been conducted using the Sociometer and EAR [WOKP10, OWK+09, OP10b, HWSM11], and some of these have also studied organisational behaviour. Although these devices have an advantage over self-reports and experience sampling methods, they are generally not carried by the users every day, and long-term studies are quite hard to achieve as it might be impractical to ask users to carry external devices for prolonged periods or users may need to be incentivised [MRG+11] which may prove costly. The applications that we designed instead target off-the-shelf smartphones. Further, in systems like EAR, it takes a lot of manual effort of social scientists to code audio files, whereas in the proposed systems classification is performed automatically. Some works [LFO+07] have deployed wireless sensor networks at the work and domestic environments to understand usage patterns (light use, temperature, motion). These approaches rely on full instrumentation of a building that may again be costly or impractical.

With respect to voice processing technologies, a survey of GMM-based speaker recognition technology can be found in [Rey02]. SoundSense [LPL+09] is a smartphone platform implemented on iPhone that detects interesting events using the microphone sensor such as riding a bus, driving a car and is capable of automatically detecting the context, which finds many potential applications in the field social sciences; similar algorithms that are complementary to ours can be added to EmotionSense to provide information about social situations to psychologists. The emotion recognition system that we implemented is close to that devised by the Brno University of Technology team for the Interspeech 2009 Emotion challenge described in [KBC09]; however, this system was not based on mobile phones. In addition to voice based emotion recognition systems, there are systems built using wearable emotion detectors [TML+08].

6.6 Conclusions

In this chapter we demonstrated using three example social applications that smartphones can be effective tools in the conduct of social studies. EmotionSense is an application for conducting passive behavioural monitoring studies and incorporates automatic emotion and speaker recognition components. We have discussed the results of the evaluation of the core application components by means of a series of benchmarks and a social study that involved 18 participants. We have also shown how the information collected by EmotionSense can be potentially used to understand the patterns of interaction and the correlation of inferred emotions with the places, groups, and activity captured from the sensors in the phone.

We then presented WorkSense, another example social sensing application that could potentially be useful in conducting behavioural monitoring studies at the workplace. The application implements collaboration and meeting inference components and we showed through real deployment that the system detects formal and informal meetings and the inferences match the actual collaborations of the users. Finally, we presented SociableSense, an application that *closes the loop* by providing real-time feedback to users on their *sociability*, the strength of their relations with colleagues, and opportunities to interact. The core of the application is a social component that models the relations of the users and ranks them. We showed through real deployment that smartphones can be potentially used as a platform to implement feedback and alert mechanisms and to test the effectiveness of these mechanisms.

7
Reflections and Future Work

In the previous chapters we presented social sensing applications and techniques to efficiently run them on smartphones. In this chapter we provide a brief overview of our thesis and the research questions that we aimed to answer. We then summarise our contributions, present limitations, and offer suggestions for future work.

Social psychology research deals with the understanding of behavioural and social aspects of users. Most social science researchers do not use smartphone sensing technology to its full potential. On the other hand, smartphone technology has been advancing at an unprecedented rate. The thesis is inspired by the ubiquitous nature, unobtrusiveness, and sensor richness of smartphones and their application in social sciences research: *smartphone sensing can be used to automatically capture the behavioural and social aspects of the user, and can be an effective tool in the conduct of social studies.*.

To support the thesis we were required to build schemes to efficiently capture and process raw sensor data, and build mobile social applications based on inferences from sensor data. However, each of these pose research questions (presented in Chapter 1, Section 1.5) that needed answering. The dissertation has supported the thesis by exploring the following closely related research threads:

1. Considering the resource limitations on mobile phones, we designed energy-efficient techniques to capture data from the sensors in the smartphone and in the user's environment, and to process the sensor data to derive inferences. These techniques enable smartphones to be able to run social sensing applications efficiently.

2. We designed techniques to infer and model the user's behaviour based on smartphone sensor data, and implemented and deployed three example social applications to demonstrate the potential of smartphone sensing in the conduct of social studies.

7.1 Summary of Contributions

In this section we reiterate the research questions that we aimed to answer and describe our contributions in detail.

Research Question 1. How can we accurately capture raw data from the sensors in smartphones in an energy-efficient way?

[**Contribution 1**] Mobile phones are equipped with many sensors such as accelerometer, Bluetooth, and microphone, and data gathering from these sensors is a fundamental step to support social sensing applications. However, since mobile phones are resource constrained, efficient techniques need to be designed to balance energy-accuracy trade-offs. We explored these trade-offs in Chapter 3. We designed an adaptive sensing framework based on the principle that interesting events should not be missed and uninteresting events can be missed. We presented a design method and an adaptive sensing scheme that adapts the sampling rate of the sensors to the context of the user to achieve energy savings while maintaining the required level of accuracy. We also presented a rules framework that allows social scientists to write rules and adapt the sampling behaviour of the system to changing external factors. We showed through evaluation using real traces that the adaptive scheme saves as much as 50% energy compared to a continuous sensing scheme while achieving a similar level of accuracy.

[**Contribution 2**] Even though the adaptive sensing scheme saves energy compared to continuous sensing, it still needs to expend energy to achieve high accuracy. Considering that modern buildings have many sensors such as motion, door sensors, cameras embedded in the infrastructure, these can be exploited to achieve further energy savings. Therefore, in Chapter 4, we designed a sensor offloading scheme that leverages the sensors in the mobile phone and in the user's environment to capture the user's data. The offloading scheme dynamically adapts to the mobility patterns of users and saves a significant amount of energy without compromising on accuracy requirements. We also showed that combining adaptive sensing with the sensing offloading scheme results in high energy savings.

Research Question 2. How can we efficiently process data captured through smartphone sensors to draw inferences about the user?

[**Contribution 3**] Based on the schemes designed in Chapters 3 and 4, data can be efficiently gathered from smartphone sensors. However, it still needs to be processed to derive inferences about the user's behaviour. Therefore, in Chapter 5, we designed a computation offloading scheme that smartly distributes the computing of a task between local mobile phone and remote cloud processing resources by considering the dimensions: energy, latency, and data sent over the network. The scheme can be adapted to the changing resources of the mobile phone by writing rules. We showed through evaluation on real traces that the computation offloading scheme selects the optimal configuration for a given task considering the requirements of the experiment designers.

Research Question 3. In what ways can smartphones be helpful in the conduct of social studies?

[**Contribution 4**] After designing schemes that can efficiently support data gathering and processing for social sensing, we focussed on the design of example social psychological applications to show using real examples the type of data that can be collected using smartphones and the analysis that can be performed on this data. In Chapter 6, we built a passive behavioural monitoring application (EmotionSense), a collaboration and interaction detection application for the workplace (WorkSense), and an application that can provide realtime feedback (SociableSense) to show the usefulness of smartphones in supporting various types of social applications. EmotionSense is a passive monitoring platform and implements two subsystems: speaker and emotion recognition, for inferring the user's emotion and speech patterns autonomously. By correlating these with other sensor data like location, co-location, and activity, fine-grained behaviour analysis can be performed. WorkSense uses the speaker recognition component of EmotionSense and the sensor offloading scheme (Chapter 4) to automatically infer the collaboration and work patterns of the users at the workplace. This application can be used to understand the impact that social activities have on the work performance of employees and can be beneficial to both the employees and the employer. Finally, we presented SociableSense, an application that provides realtime feedback and alerts to users to encourage them to be more sociable. The application models the relations of the users based on the microphone (interactions) and Bluetooth (colocation) sensor data.

These contributions in terms of efficiently supporting social sensing applications on smartphones and designing models to capture the user's behaviour from sensor data supported our thesis: smartphone sensing can be used to automatically capture the behavioural and social aspects of the user, and can be an effective tool in the conduct of social studies.

7.2 Limitations

In this section we present the limitations of the proposed schemes and the social applications.

- The adaptive sensing scheme is applicable to capturing data from a sensor only if the events inferred from the sensor stream can be classified to interesting or not by the application. Further, it may not be suitable to sensor streams with low-frequency events, i.e., events that happen rarely, as there is a high chance of missing them especially when the probability of sensing is lower.

- The sensing offloading scheme can only be used in environments where a sensing infrastructure is available. Further, it also assumes the availability of a discovery and access service (for example, a web-based API) to find and utilise the remote sensors.
- The computation offloading scheme requires the values of the dimensions (such as energy, latency) to be pre-configured before executing the decision engine. Further, the rules framework that can be used to dynamically adapt the behaviour of the decision engine, lacks a way to find the conflicting rules defined by users.
- The social sensing applications that we presented in the previous chapter are some examples to show the type of data that can be collected using off-the-shelf smartphones and the type of analysis that can be performed on this collected data. In order to confirm the accuracy of findings and inferences from these social studies, more focussed and/or large scale studies in collaboration with social scientists are required.

7.3 Future Directions

The rapid growth of the number of smartphone users is likely to continue for many years [YAH]. As sensor technology advances there will be many sensors added to commodity smartphones [LML$^+$10]. Sensors in modern mobile phones can be used to capture the behaviour of users automatically and accurately as demonstrated in this dissertation. One potential use of the schemes and classifiers presented in this dissertation could be to understand the bias introduced in various experience sampling studies as shown in our recent work [LRMR13]. In this section we offer suggestions for future work.

7.3.1 Open Sensing Infrastructure Access

We have shown in this dissertation that adaptive sensing balances the energy-accuracy trade-offs of social sensing applications. However, energy still needs to be expended to achieve higher accuracy. As shown in Chapter 4, leveraging the environmental sensors is an approach that results in higher accuracy while saving a significant amount of energy. This is a key mechanism that can support the long term deployment of mobile social applications. Considering the benefit to employees and employers, there is a case for organisations to open access to the sensing infrastructure. Further, there is also a need to create open standards for generalising the way in which mobile devices can access building infrastructure sensors. Given that these can be charged to users through billing or through advertising, there is a need for re-thinking the business case in this area.

7.3.2 Behavioural Interventions

In the domain of social sensing an area that can be explored and which has not fully exploited smartphone sensing is behavioural interventions. We have shown that mobile phones could be used to provide realtime feedback through SociableSense in Chapter 6. They can also be used in more varied scenarios, for example, in assisting users to quit cigarette smoking. Another example is to help in reducing stress through intervention when stress level increases (can be detected using heart rate monitors and/or a microphone sensor [LRC$^+$12]). However, the detection of these activities, for example, cigarette smoking, is limited by available sensors in the phone. It might be possible to use alternative sensors like accelerometers to identify smoking activity based on the actions or activity patterns. Other behavioural interventions include helping users reduce their carbon footprint by detecting their energy usage, and to reduce water consumption. Since many users always carry phones and more importantly, regularly interact with them, they represent a perfect platform for implementing persuasive techniques and interventions.

7.3.3 Evolving Classifiers

Emotion and mood recognition find applications in many domains including behavioural psychology, healthcare, and stress detection. Furthermore, identifying mood can also be exploited for commercial purposes such as delivering mood appropriate advertisements. We have shown the feasibility of emotion recognition using mobile phones in Chapter 6. However, detecting emotions is a complex task as there are many factors to be considered such as noise levels and the way emotions are expressed by different groups (people of different cultural background, age groups, etc). These factors directly influence the accuracy of the classification. Further, it is not always possible to collect training data for each of these categories. A logical step ahead in improving the accuracy is to design evolving classifiers that can use the real-time recorded data to fine-tune and adapt classifiers to the user. A recent work [LRC$^+$12] has shown that using classifiers that adapt to the user is an effective method. More work in this direction could lead to scalable and accurate emotion detection classifiers.

* * *

Smartphones are already one of the most popular electronic devices. Phone vendors have been adding a variety of sensors to already sensor-rich off-the-shelf smartphones. The availability of sensors in phones, especially GPS and magnetometer sensors has been exploited by many popular mobile applications like Foursquare[1] and Google Latitude[2] that are used by millions of users. There is also an increasing trend in use of other sensors, such as the accelerometer and microphone sensors, for activity and conversation detection in mobile applications such as CenceMe[3]. Further, some recent applications have also aimed at exploring the use of mobile phones and their sensors for behavioural and health sciences, like BeWell[4] and ginger.io[5].

Smartphone battery technology is a major limiting factor for the growth of mobile applications, especially social sensing applications. This dissertation has demonstrated that by using smart schemes for sensing and processing, it is possible to run social sensing applications efficiently on smartphones. The contribution of this dissertation is to have paved the way for many more interesting applications based on smartphones in the field of social and behavioural sciences.

[1]https://foursquare.com/
[2]http://www.google.com/latitude/
[3]http://itunes.apple.com/us/app/cenceme/id284953822?mt=8
[4]https://play.google.com/store/apps/details?id=org.bewellapp
[5]http://ginger.io/

A
Adaptive Sensing API

In this appendix we present the design of APIs for the adaptive sensing framework presented in Chapter 3. These APIs can be used by the sensor sampling components of mobile applications to efficiently capture data from the sensors in a smartphone. We implemented these APIs using Java on the Android 2.3.3 platform.

A.1 API

In the API, we refer to the classes that sense data from the sensors as activity trackers as they track the activity of the users such as physical activity, co-location activity, and speech activity etc.

`Class: AdaptiveSensing`

`public AdaptiveSensing getAdaptiveSensing()`

The adaptive sensing component controls the interval between consecutive sampling windows of the activity trackers. For this, each activity tracker should implement a listener interface (`AdaptiveSensingListener`) and register with the `AdaptiveSensing`, which is a singleton class and can be obtained using the above method.

```
public int registerActivityTracker(AdaptiveSensingListener li-
stener, ActivityTrackerInfo trackerInfo, int sensingCriteria,
<optional> InterestClassifier classifier)
```

An activity tracker that requires adaptive sensing functionality should implement a listener interface `AdaptiveSensingListener` and register its instance with the singleton instance of `AdaptiveSensing`, so that it can control the sampling interval of the tracker. This can be done at the system start-up. The `ActivityTrackerInfo` object has information about the tracker such as sensor type and classifier type. `sensingCriteria` specifies the requirements of the activity tracker in terms of accuracy. `InterestClassifier` is an optional parameter that can be passed in the registration process so that it can be used to classify the sensor events from the activity tracker to interesting or not. If the optional parameter is not specified, then the default classifier for the activity tracker will be used. The return type of this method is an identifier that uniquely identifies the registration and is required for all subsequent communication with the adaptive sensing instance. Details about these classes/interfaces will be presented further in this section.

The implementation of an `ActivityTracker` should consider a sleep time between any two consecutive sensor sampling windows, and this sleep time should be updated in the method `AdaptiveSensingListener.onSamplingWindowInteralChanged()` called by the `AdaptiveSensing` component.

`public boolean deregisterActivityTracker(int identifier)`

An activity tracker can also control its sampling rate independent of the adaptive sensing. If an activity tracker is already registered and no longer requires adaptive sampling, it can then unregister from the adaptive sensing component.

`public void logSensedEvent(int identifier, SensorEvent sensorEvent)`

This method informs the adaptive sensing component about the last sensed event. The `AdaptiveSensing` then classifies this `sensorEvent` to interesting or not and then updates the sampling interval of the tracker accordingly.

`Interface: AdaptiveSensingListener`

`public void onSamplingWindowIntervalChanged(long millis)`

Called by the adaptive sensing component to set the sampling interval of the registered sensor tracker. The parameter `millis` is the sleep interval in milliseconds between two consecutive sampling windows. In the tracker's listener implementation, this method should not be used to perform any intensive processing as this will block the adaptive sensing component notification thread, and should only be used to receive the sleep interval value and return the call.

`Class: ActivityTrackerInfo`

`public int getSensorType()`

This method returns the sensor type of the activity tracker. This can be `SensorType.ACC-ELEROMETER`, `SensorType.MICROPHONE` etc. The method is required as the sampling rate variation is dependent on the type of the sensor, as shown in Chapter 3.

`public int getClassifierType()`

This method returns the classifier type of the activity tracker. This can be `ClassifierTy-pe.BASIC_MOVEMENT`, `ClassifierType.SPEECH_RECOGNITION` etc. The method is required as the sampling rate variation is dependent on the type of the classifier too.

`Class: SensingCriteria`

This class provides various sensing criterion options. The actual sampling interval for a sensing criterion, and the quantification of accuracy is based on the α_I and α_D values presented in Chapter 3.

`public int SENSOR_ACCURACY_HIGH`

This criterion indicates high sensor sampling rate to achieve high accuracy, and possibly high power consumption.

`public int SENSOR_ACCURACY_MEDIUM`

This criterion indicates medium sensor sampling rate to achieve medium accuracy. This criterion achieves balanced energy-accuracy trade-offs.

`public int SENSOR_ACCURACY_LOW`

This criterion indicates low sensor sampling rate to achieve energy savings, however, it might be at the cost of reduced accuracy.

`Interface: InterestClassifier`

`public boolean isInteresting(SensorEvent event)`

This method returns a boolean value indicating whether the input `SensorEvent` is interesting or not (unmissable or missable event, presented in Chapter 3, Section 3.2). The classification of an event to interesting or uninteresting is dependent on the application requirements. The interface is provided in order for an `ActivityTracker` to define specialised interest classifiers if needed. By default the adaptive sensing component includes general-purpose interest classifiers for a set of activity trackers (presented in Chapter 3, Section 3.5). An `InterestClassifier` implementation is provided per `ActivityTracker` and is based on the events generated by the latter. The current implementation provides a static mapping of events to interesting or uninteresting (e.g., for microphone sensor, sound event is interesting and silence event is uninteresting). The `InterestClassifier` for a particular activity tracker will have knowledge about the type of events (returned by `SensorEvent.getEventType()`) generated by the tracker.

`Interface: SensorEvent`

`public int getEventType()`

This method returns the event type of the sensor event. Event type is an integer value that represents the type of this event, for example, user is walking, user is speaking, and user is at the workplace etc.

`public Object getEventData()`

This method returns an object that contains the event data. Since the type of data is dependent on the sensor and classifier, the implementing class should return the appropriate type of object.

`public long getTimeStamp()`

This method returns the time (in milliseconds) at which the event was detected.

`public String toString()`

This method returns a string representation (human readable) of the sensor event object. For example, for movement classifier this could be "walking" or "running".

`Class: Sensor Type`

Sensor type class defines various sensors in mobile phones that are supported by the adaptive sensing framework. If the sensor used by an activity tracker is not defined in this class, then the adaptive sensing cannot support it.

`public static int ACCELEROMETER;`

This represents the 3-axis accelerometer sensor embedded in the phone. Generally used for physical activity recognition of the user.

`public static int MICROPHONE;`

This represents the microphone sensor embedded in the phone. Generally used for noise and speech detection.

`public static int GPS;`

This represents the GPS sensor (for location detection) embedded in the phone.

`public static int BLUETOOTH;`

This represents the Bluetooth sensor embedded in the phone. Typically used for indoor localisation and co-location detection.

`Class: Classifier Type`

Classifier type class defines various classifiers supported. If the classifier used by an activity tracker is not defined in this class, then the adaptive sensing cannot support it.

`public static int PHYSICAL_ACTIVITY;`

This represents a physical activity classifier such as movement classifier described in Chapter 3, Section 3.5 that can classify raw accelerometer data to moving or stationary states.

`public static int SPEAKER_IDENTIFICATION;`

This represents a speaker identification classifier typically based on machine learning techniques and intensive in processing, for example, speaker recognition technique presented in Chapter 6, Section 6.2.

`public static int LOCATION_DETECTION;`

This represents a location detection classifier based on one or more of the following: GPS, Wi-Fi, Cell towers, and Bluetooth devices etc.

A.2 Sample Code

The following sample code provides a high level example of how the adaptive sensing API can be used by the developers of sensor trackers and mobile sensing applications using the Android Java programming language. In this example, we present an activity tracker that detects physical activity of the user such as: running, walking, driving etc. and show how it should use the adaptive sensing API.

```java
public class RunningTracker implements
            ActivityTracker, AdaptiveSensingListener, Runnable
{

  // sleep between sensor sampling windows
  private long sensorSleepInterval;
  // adaptive sensor sampling
  private AdaptiveSensing adaptiveSensing;
  // activity tracker info: sensor type, classifier
  private ActivityTrackerInfo trackerInfo;
  // boolean to indicate whether to stop the tracker or not
  private boolean stopTracker = false;
  // identifier returned by the adaptive sensing component
  private int identifier;

  public RunningTracker()
  {
      // get the instance of the adaptive sensing service
      adaptiveSensing = AdaptiveSensing.getAdaptiveSensing();
```

```
    // create tracker info object
    trackerInfo = new ActivityTrackerInfo(SensorType.ACCELEROMETER,
              ClassifierType.PHYSICAL_ACTIVITY);
    // register with the adaptive sensing
    identifier = adaptiveSensingService.register(this, trackerInfo,
              SensingCriteria.SENSOR_ACCURACY_HIGH);
}

/* sensor sampling: sense and sleep cycles can be
   implemented based on the Java runnable interface or using
   the Android AlarmManager;  in this example, we base
   the implementation on the Java runnable interface */
public void run()
{
    while (! stopTracker)
    {
        // capture data from the accelerometer sensor
        SensorEvent sensorEvent = sense();
        // log the event to the adaptive sensing instance
        adaptiveSensing.logSensedEvent(identifier, sensorEvent);
        // log to db or process
        process(sensorEvent);
        // sleep
        sleep(sensorSleepInterval);
      }
}

/* this method is called by the adaptive sensing
   component when a new sleep interval is calculated.
   developers should not perform any intensive computing
   in this method, as it will block the adaptive sensing
   notification thread */
public void onSamplingWindowIntervalChanged (long interval)
{
  // update sleep interval
  sensorSleepInterval = interval;
}

private void sense()
{
    // capture sensor data using the Android SensorManager
}
```

```

    private void process(SensorEvent sensorEvent)
    {
        // process sensor data
    }

    // called on shutdown
    public void shutdown()
    {
        stopTracker = true;
        adaptiveSensing.dereigsterActivityTrackerListener(identifier);
    }

}
```

B
Computation Offloading API

In this appendix we present the design of APIs for the computation offloading scheme presented in Chapter 5. We implemented these APIs using the Java programming language on the Android platform. These APIs can be used by mobile applications to use the services of the computation offloading scheme to efficiently utilise the local phone and remote computing resources. As discussed in Chapter 5, each classification task can be divided into one or more subtasks, for example, speaker identification can be divided into converting audio file to coefficients file, and comparing this with the existing speaker models (as described in Chapter 6, Section 6.2). Each subtask can have one or more subtask versions each implementing a computation paradigm. For example, a subtask version of a subtask could be to perform model comparison for speaker identification locally on the phone and another could be to perform the same remotely in the cloud.

Based on the requirements of the task and the weights given to the dimensions: energy, latency, and data traffic (as described in Chapter 5, Section 5.2), the decision engine decides for each subtask, which subtask version to execute. The programmer can specify the properties and requirements of a task either in an XML configuration file or programmatically using the APIs. An example XML file is shown in Figure B.1 for a speaker identification task (task id: `speaker_identification`), which consists of two subtasks: extracting features (subtask id: `feature_extraction`) and model comparison (subtask id: `model_comparison`), each of which has several versions (based on local or remote computation). Pre-filtering requirements could be specified for the task in terms of privacy and maximum latency in the `task` element.

```
<?xml version="1.0" encoding="UTF-8"?>
<tasks xmlns="Cambridge:CO"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="Cambridge:CO tasks.xsd">
    <task id="speaker_identification">
        <max-latency>100000</max-latency>
        <min-privacy-level>low</min-privacy-level>
        <subtask id="feature_extraction" position="1">
            <subtask-version version="1">
                <computation-model>remote</computation-model>
                <input-size>2048</input-size>
                <output-size>128</output-size>
                <energy>10</energy>
                <latency>500</latency>
            </subtask-version>
            <subtask-version version="2">
                <computation-model>local</computation-model>
                <input-size>2048</input-size>
                <output-size>128</output-size>
                <energy>25</energy>
                <latency>2000</latency>
            </subtask-version>
        </subtask>
        <subtask id="model_comparison" position="2">
            <subtask-version version="1">
                <computation-model>remote</computation-model>
                <input-size>128</input-size>
                <output-size>128</output-size>
                <energy>20</energy>
                <latency>600</latency>
            </subtask-version>
            <subtask-version version="2">
                <computation-model>local</computation-model>
                <input-size>128</input-size>
                <output-size>128</output-size>
                <energy>40</energy>
                <latency>12000</latency>
            </subtask-version>
        </subtask>
    </task>
</tasks>
```

Figure B.1: Sample configuration XML file for the specification of tasks.

B.1 API

`Class: ComputationDistribution`

This class provides access to the decision engine of the computation distribution framework. It provides the following functions (only public functions are listed).

`public static ComputationDistribution getComputationDistribution()`

`ComputationDistribution` is a singleton class and the instance of its object can be obtained using the above method.

`public Task loadTask(String taskName)`

Reads the task and all subtask variants from the XML configuration file and returns a `Task` object.

`public int registerTask(Task task)`

Before the first execution, a task has to be registered with the framework using this function. This method returns an `identifier`, which should be passed in all the subsequent task-related communication with the `ComputationDistribution` component.

`public int unregisterTask(int identifier)`

After the last execution of the task, unregistering removes the task from the task list. The input parameter is the identifier provided by the `registerTask()` function.

`public List<SubtaskVersion> whichSubtaskVersions(int identifier)`

This executes the decision engine (Chapter 5, Algorithm 1) and returns an array of subtask versions (one subtask version for each subtask) to execute.

`public Object executeTask(List<SubtaskVersion> subtaskVersions, int identifier, Object input)`

Executes the task by executing each subtask version in the array `subtaskVersions` using the provided `input` object as the input to the first subtask, the output of the first subtask as the input to the second subtask, and so on. The output of the last subtask is returned.

`Class: Task`

This class represents a task. Instance of a task can be created by the `loadTask()` function of `ComputationDistribution` or can be created and populated manually. The class provides getters and setters for the pre-filters: privacy level and maximum latency. The privacy level is used to specify if the remote computation can be used or not. If the privacy level is set to high, then only local computation is used otherwise both local and cloud computation models are considered. If the latency incurred by the local or cloud models is more than the maximum latency value, then that model is not considered for computing the task.

`public Task(String id, PrivacyLevel minPrivacyLevel, long maxLatency, List<Subtask> subtasks)`

Constructor: Create a task with given id, pre-filter values, and list of subtasks.

`Class: Subtask`

This class represents a subtask.

`public Subtask(List<SubtaskVersion> subtaskVersions)`

Constructor: Create a subtask with given list of subtask versions.

`Abstract Class: SubtaskVersion`

Each subtask version must extend `SubtaskVersion` and implement the `execute` function. The class further provides getters and setters for the subtask version properties id, energy, latency, computation model, input size, and output size.

`public SubtaskVersion(int id, long energy, long latency, long inputSize, long outputSize, ComputationModel cm)`

Constructor: create a subtask version with provided id, expected energy (Joules), latency (milliseconds), input data size (Bytes), output data size (Bytes), and computation model.

`abstract Object execute(Object input)`

The `execute` function must be implemented by each subtask version. The `executeTask()` method in `ComputationDistribution` calls this method.

`Enum: public enum PrivacyLevel LOW, HIGH`

Defines the privacy levels, `LOW`: input and output data can be sent over the network; `HIGH`: input and output data cannot be sent over the network.

`Enum: public enum ComputationModel LOCAL, REMOTE`

Enumeration class representing local and remote computation models.

B.2 Sample Code

The following sample code provides a high level example of how the computation offloading API can be used by the developers of smartphone sensor trackers and mobile sensing applications. In this example, a thread uses an audio tracker to capture microphone data periodically (could be through the adaptive sensing API) and then uses `ComputationDistribution` class to decide which subtask versions to use to compute the task and then executes them.

```
/**
 * This class implements a thread, which repeatedly
 * listens to the microphone for 5 seconds and then
 * performs speaker identification.
 */
public class SpeakerRecognitionThread extends Thread
{

  // task id (as defined in the XML)
  private String taskId = "speaker_identification";
  // how long to record before identification
  private long listenInterval = 5000;
  // we continue speaker identification until false
  public boolean running = true;

  // the computation distribution
  private ComputationDistribution cd;
  // the task
  private Task task;
  // id assigned by the computation distribution
  private int id;

  public void run()
  {
    // get the instance of the computation distribution
    cd= ComputationDistribution.getComputationDistribution();
    // load task definitions from the XML file
    task = cd.loadTask(taskId);
    // register the loaded task with the computation distribution
    id = cd.registerTask(task);

    // repeatedly execute speaker identification
    while(running)
    {
      // wait for a recorded event from audio tracker.
      // in this example, we assume the tracker logs the event
      // to a queue when an audio sample is recorded, but other
      // methods can also be used
      synchronized (queue)
      {
        queue.wait();
      }
      // get the recorded audio file from tracker
      byte[] input = queue.remove();

      // find which subtask versions to compute
      SubtaskVersion[] subtaskVersions = cd.whichSubtaskVersions(id);

      // execute the speaker identification
      cd.executeTask(subtaskVersions, id, input);
    }

    // unregister the task from the computation distribution
    cd.unregisterTask(id);
  }

}
```

Bibliography

[ABS05] Ernesto Arroyo, Leonardo Bonanni, and Ted Selker, *Waterbot: exploring feedback and persuasive techniques at the sink*, Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '05), ACM, 2005.

[ACRC09] Martin Azizyan, Ionut Constandache, and Romit Roy Choudhury, *Surround-Sense: Mobile Phone Localization via Ambience Fingerprinting*, Proceedings of the International Conference on Mobile Computing and Networking (MobiCom'09), ACM, 2009.

[ALC06] Daniel Ashbrook, Kent Lyons, and James Clawson, *Capturing experiences anytime, anywhere*, IEEE Pervasive Computing **5** (2006), 8–11.

[ALM] *Android location manager*, http://developer.android.com/reference/android/location/LocationManager.html.

[AND] *Android*, http://www.android.com/.

[ASSC02] Ian Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci, *A survey on sensor networks*, IEEE Communications Magazine **40** (2002), no. 8, 102–114.

[AUC] *Audacity*, http://audacity.sourceforge.net/.

[BAB$^+$10] Nilanjan Banerjee, Sharad Agarwal, Paramvir Bahl, Ranveer Chandra, Alec Wolman, and Mark Corner, *Virtual Compass: Relative Positioning to Sense Mobile Social Interactions*, Proceedings of the International Conference on Pervasive Computing (Pervasive'10), LCNS Springer, 2010.

[BAPH09] Fehmi Ben Abdesslem, Andrew Phillips, and Tristan Henderson, *Less is more: energy-efficient mobile sensing with senseless*, Proceedings of the 1st ACM Workshop on Networking, Systems, and Applications for Mobile Handhelds (MobiHeld'09), ACM, 2009.

[BB01] Lisa Feldman Barrett and Daniel J. Barrett, *An Introduction to Computerized Experience Sampling in Psychology*, Social Science Computer Review **19** (2001), no. 2, 175–185.

[BBV09] Niranjan Balasubramanian, Aruna Balasubramanian, and Arun Venkataramani, *Energy consumption in mobile phones: a measurement study and implications for network applications*, Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference (IMC'09), ACM, 2009.

[BDR03] Niall Bolger, Angelina Davis, and Eshkol Rafaeli, *Diary methods: Capturing life as it is lived*, Annual Review of Psychology **54** (2003), no. 1, 579–616.

[BEH$^+$06] J. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M. B. Srivastava, *Participatory sensing*, Proceedings of First Workshop on World-Sensor-Web: Mobile Device Centric Sensory Networks and Applications, 2006.

[BGLL08] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre, *Fast unfolding of communities in large networks*, Journal of Statistical Mechanics: Theory and Experiment (2008).

[BGR12] *1 million android devices activated each day, 400 million total*, 2012, http://www.bgr.com/2012/06/27/1-million-android-devices-activated-each-day-400-million-total/.

[BH75] Gordon H. Bower and Ernest R. Hilgard, *Theories of Learning*, Prentice-Hall, Inc., 1975.

[Bis06] Christopher M. Bishop, *Pattern recognition and machine learning (information science and statistics)*, Springer, 2006.

[Bis07] C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2007.

[Bla86] P. H. Blaney, *Affect and memory: A review*, Psychological Bulletin (1986), 229–246.

[BNJ11] Rajesh Krishna Balan, Khoa Xuan Nguyen, and Lingxiao Jiang, *Real-time trip information service for a large taxi fleet*, Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services (MobiSys'11), ACM, 2011.

[BOH83] Van Bezooijen, Otto, and Heenan, *Recognition of vocal expressions of emotion: A three-nation study to identify universal characteristics*, Journal of Cross-Cultural Psychology **14** (1983), 387–406.

[Bon03] P. Bonato, *Wearable sensors/systems and their impact on biomedical engineering*, Engineering in Medicine and Biology Magazine, IEEE **22** (2003), no. 3, 18 –20.

[BP00] P. Bahl and V.N. Padmanabhan, *Radar: An in-building RF-based user location and tracking system*, Proceedings of the IEEE Conference on Computer Communications (INFOCOM'00), IEEE, 2000.

[BR07] M. Botts and A. Robin, *Opengis sensor model language (sensorml) implementation specification*, 2007.

[BRC$^+$07] Nilanjan Banerjee, Ahmad Rahmati, Mark Corner, Sami Rollins, and Lin Zhong, *Users and Batteries: Interactions and Adaptive Energy Management in Mobile Systems*, Proceedings of the International Conference on Ubiquitous Computing (UbiComp'07), ACM, 2007.

[BS96] R. Banse and K. R. Scherer, *Acoustic profiles in vocal emotion expression*, Journal of Personality and Social Psychology **70** (1996), no. 3, 614–636.

[BSPO03] Rajesh Krishna Balan, Mahadev Satyanarayanan, So Young Park, and Tadashi Okoshi, *Tactics-based remote execution for mobile computing*, Proceedings of the 1st International Conference on Mobile Systems, Applications and Services (MobiSys '03), ACM, 2003.

[Bur95] Ronald Burt, *Structural Holes: The Social Structure of Competition*, Harvard University Press, 1995.

[BYAH06] Michael Buettner, Gary V. Yee, Eric Anderson, and Richard Han, *X-mac: a short preamble mac protocol for duty-cycled wireless sensor networks*, Proceedings of the 4th International Conference on Embedded Networked Sensor Systems (SenSys '06), ACM, 2006.

[CAHS05] Qing Cao, Tarek Abdelzaher, Tian He, and John Stankovic, *Towards optimal sleep scheduling in sensor networks for rare-event detection*, Proceedings of the International Symposium on Information Processing in Sensor Networks (IPSN '05), IEEE Press, 2005.

[CAN11] *Smart phones overtake client PCs*, 2011, http://www.canalys.com/newsroom/smart-phones-overtake-client-pcs-2011.

[CBC+08] Tanzeem Choudhury, Gaetano Borriello, Sunny Consolvo, Dirk Haehnel, Beverly Harrison, Bruce Hemingway, Jeffrey Hightower, Predrag "Pedja" Klasnja, Karl Koscher, Anthony LaMarca, James A. Landay, Louis LeGrand, Jonathan Lester, Ali Rahimi, Adam Rea, and Danny Wyatt, *The Mobile Sensing Platform: An Embedded Activity Recognition System*, IEEE Pervasive Computing **7** (2008), no. 2, 32–41.

[CBC+10] Eduardo Cuervo, Aruna Balasubramanian, Dae-ki Cho, Alec Wolman, Stefan Saroiu, Ranveer Chandra, and Paramvir Bahl, *MAUI: Making Smartphones Last Longer with Code Offload*, Proceedings of the International Conference on Mobile Systems, Applications, and Services (MobiSys'10), ACM, 2010.

[CCC+09] Meng-Chieh Chiu, Shih-Ping Chang, Yu-Chen Chang, Hao-Hua Chu, Cheryl Chia-Hui Chen, Fei-Hsiu Hsiao, and Ju-Chun Ko, *Playful bottle: a mobile social persuasion system to motivate healthy water intake*, Proceedings of the 11th International Conference on Ubiquitous Computing (Ubicomp'09), ACM, 2009.

[CEL+08] Andrew T. Campbell, Shane B. Eisenman, Nicholas D. Lane, Emiliano Miluzzo, Ronald A. Peterson, Hong Lu, Xiao Zheng, Mirco Musolesi, Kristóf Fodor, and Gahng-Seop Ahn, *The rise of people-centric sensing*, IEEE Internet Computing (2008).

[CEP+07] Jinhai Cai, D. Ee, Binh Pham, P. Roe, and Jinglan Zhang, *Sensor network for the monitoring of ecosystem: Bird species recognition*, Proceedings of the 3rd International Conference on Intelligent Sensors, Sensor Networks and Information (ISSNIP'07), 2007.

[CGS+09] Ionut Constandache, Shravan Gaonkar, Matt Sayler, Romit Roy Choudhury, and Landon Cox, *EnLoc: Energy-Efficient Localization for Mobile Phones*, Proceedings of the International Conference on Computer Communications (INFOCOM'09), IEEE, 2009.

[CIM+11] Byung-Gon Chun, Sunghwan Ihm, Petros Maniatis, Mayur Naik, and Ashwin Patti, *CloneCloud: Elastic Execution between Mobile Device and Cloud*, Proceedings of the ACM European Conference on Computer Systems (EuroSys'11), ACM, 2011.

[CKK$^+$08] C. Cornelius, A. Kapadia, D. Kotz, D. Peebles, M. Shin, and N. Triandopoulos, *AnonySense: privacy-aware people-centric sensing*, Proceedings of the International Conference on Mobile Systems, Applications, and Services (MobiSys'08), ACM, 2008.

[CL87] M. Csikszentmihalyi and R. Larson, *Validity and reliability of the experience-sampling method*, Journal of Nervous & Mental Disease (1987), 526–536.

[CLL$^+$12] Yohan Chon, Nicholas D. Lane, Fan Li, Hojung Cha, and Feng Zhao, *Automatically characterizing places with opportunistic crowdsensing using smartphones*, Proceedings of the International Conference on Ubiquitous Computing (Ubicomp'12), ACM, 2012.

[CMP00] B. Clarkson, K. Mase, and A. Pentland, *Recognizing user context via wearable sensors*, Proceedings of the Fourth International Symposium on Wearable Computers (ISWC'00), 2000.

[COS12] *Comscore*, 2012, http://www.comscore.com/Press_Events/Press_Releases/2012/5/Introducing_Mobile_Metrix_2_Insight_into_Mobile_Behavior.

[CP03] Tanzeem Choudhury and Alex Pentland, *Sensing and Modeling Human Networks using the Sociometer*, Proceedings of the International Symposium on Wearable Computers (ISWC'03), 2003.

[CSR06] W.M. Campbell, D. Sturim, and D.A. Reynolds, *Support vector machines using GMM-supervectors for speaker verification*, IEEE Signal Processing Letters **13** (2006), 308–311.

[CW88] Lee A. Clark and David Watson, *Mood and the mundane: Relations between daily life events and self-reported mood*, Journal of Personality and Social Psychology **54.2** (1988), 296–308.

[Dar10] Waltenegus Dargie, *Context-Aware Computing and Self-Managing Systems*, CRC Press, 2010.

[DBB11] *Infographic: Mobile Statistics*, 2011, http://www.digitalbuzzblog.com/2011-mobile-statistics-stats-facts-marketing-infographic/.

[DC09] Linda Deng and Landon P. Cox, *Livecompare: grocery bargain hunting through participatory sensing*, Proceedings of the 10th workshop on Mobile Computing Systems and Applications (HotMobile'09), ACM, 2009.

[DEM$^+$10] Vladimir Dyo, Stephen A. Ellwood, David W. Macdonald, Andrew Markham, Cecilia Mascolo, Bence Pásztor, Salvatore Scellato, Niki Trigoni, Ricklef Wohlers, and Kharsim Yousef, *Evolution and sustainability of a wildlife monitoring sensor network*, Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems (SenSys'10), ACM, 2010.

[DFN$^+$09] Nigel Davies, Adrian Friday, Peter Newman, Sarah Rutlidge, and Oliver Storz, *Using bluetooth device names to support interaction in smart environments*, Proceedings of the International Conference on Mobile Systems, Applications, and Services (MobiSys'09), ACM, 2009.

[DGP12] Trinh Minh Tri Do and Daniel Gatica-Perez, *Contextual conditional models for smartphone-based human mobility prediction*, Proceedings of the International Conference on Ubiquitous Computing (Ubicomp'12), ACM, 2012.

[DHJT$^+$10] Stephen Dawson-Haggerty, Xiaofan Jiang, Gilman Tolle, Jorge Ortiz, and David Culler, *sMAP: a simple measurement and actuation profile for physical information*, Proceedings of the ACM Conference on Embedded Network Sensor Systems (SenSys'10), 2010.

[Don06] D.L. Donoho, *Compressed Sensing*, IEEE Transactions on Information Theory **52** (2006), no. 4, 1289–1306.

[DOO08] Rodrigo De Oliveira and Nuria Oliver, *Triplebeat: enhancing exercise performance with persuasion*, Proceedings of the 10th International Conference on Human Computer Interaction with Mobile Devices and Services (MobileHCI'08), ACM, 2008.

[ELP$^+$12] Christos Efstratiou, Ilias Leontiadis, Marco Picone, Kiran K. Rachuri, Cecilia Mascolo, and Jon Crowcroft, *Sense and sensibility in a pervasive world*, Proceedings of the 10th International Conference on Pervasive Computing (Pervasive'12), 2012.

[ENG12] *Nielsen: Over 50 percent of us mobile users own smartphones*, 2012, http://www.engadget.com/2012/05/07/nielsen-smartphone-share-march-2012/.

[FBR98] L. Feldman Barrett and J.A. Russell, *Independence and bipolarity in the structure of current affect.*, Journal of Personality and Social Psychology **74** (1998), 967–984.

[FCC$^+$07] 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*, Proceedings of the International Conference on Mobile Systems, Applications, and Services (MobiSys'07), ACM, 2007.

[FCC11] D. Fisher, K. Chang, and J. Canny, *Ammon: A speech analysis library for analyzing affect, stress, and mental health on mobile phones*, Proceedings of the International Workshop on Sensing Applications on Mobile Phones (PhoneSense'11), ACM, 2011.

[FDK$^+$09] Jon Froehlich, Tawanna Dillahunt, Predrag Klasnja, Jennifer Mankoff, Sunny Consolvo, Beverly Harrison, and James A. Landay, *Ubigreen: investigating a mobile tool for tracking and supporting green transportation habits*, Proceedings of the 27th International Conference on Human Factors in Computing Systems (CHI '09), ACM, 2009.

[FF00] Friedrich Foerster and Jochen Fahrenberg, *Motion pattern and posture: Correctly assessed by calibrated accelerometers*, Behavior Research Methods **32** (2000), 450–457.

[Fis65] Peter C. Fishburn, *Independence in Utility Theory with Whole Product Sets*, Operations Research **13** (1965), no. 1, 28–45.

[Fis68] ______, *Utility Theory*, Management Science **14** (1968), 335 – 378.

[FKK11] R. Friedman, A. Kogan, and Y. Krivolapov, *On power and throughput tradeoffs of wifi and bluetooth in smartphones*, Proceedings of the IEEE Conference on Computer Communications (INFOCOM'11), IEEE, 2011.

[FMPP07] J. Fahrenberg, M. Myrtek, K. Pawlik, and M. Perrez, *Ambulatory assessment - monitoring behavior in daily life settings. a behavioral-scientific challenge for psychology*, European Journal of Psychological Assessment **23** (2007), 206–213.

[FMT$^+$99] J. Farringdon, A.J. Moore, N. Tilbury, J. Church, and P.D. Biemond, *Wearable sensor badge and sensor jacket for context awareness*, Proceedings of the Third International Symposium on Wearable Computers (ISWC'99), 1999.

[FPS02] J. Flinn, S. Park, and M. Satyanarayanan, *Balancing performance, energy, and quality in pervasive computing*, Proceedings of the International Conference on Distributed Computing Systems (ICDCS'02), IEEE, 2002.

[FR91] T. Fruchterman and E. Reingold, *Graph drawing by force-directed placement*, Software: Practice and Experience (1991).

[GFJ+09] Omprakash Gnawali, Rodrigo Fonseca, Kyle Jamieson, David Moss, and Philip Levis, *Collection tree protocol*, Proceedings of the ACM Conference on Embedded Network Sensor Systems (SenSys'09), ACM, 2009.

[GIZ12] *The Most Popular iPhone and iPad Apps of All Time*, 2012, http://gizmodo.com/5890602/the-most-popular-iphone-and-ipad-apps-of-all-time.

[GL11] Ben Greenstein and Brent Longstaff, *Followme: Enhancing mobile applications with open infrastructure sensing*, Proceedings of the 12th Workshop on Mobile Computing Systems and Applications (HotMobile'11), ACM, 2011.

[GLC+08] Shravan Gaonkar, Jack Li, Romit Roy Choudhury, Landon Cox, and Al Schmidt, *Micro-blog: sharing and querying content through mobile phones and social participation*, Proceedings of the International Conference on Mobile Systems, Applications, and Services (MobiSys'08), ACM, 2008.

[GMP] *Google maps*, http://maps.google.co.uk/.

[Gol04] Scott A. Golder, *The keep-in-touch phone: a persuasive telephone for maintaining relationships*, CHI'04 Extended Abstracts on Human Factors in Computing Systems (CHI EA'04), ACM, 2004.

[Goo61] L.A. Goodman, *Snowball sampling*, The Annals of Mathematical Statistics **32** (1961), 148–170.

[Her90] H. Hermansky, *Perceptual linear predictive (PLP) analysis of speech*, Journal of the Acoustical Society of America **87** (1990), no. 4.

[HGDW12] Timothy W. Hnat, Erin Griffiths, Ray Dawson, and Kamin Whitehouse, *Doorjamb: Unobtrusive room-level tracking of people in homes using doorway sensors*, Proceedings of the ACM Conference on Embedded Network Sensor Systems (SenSys'12), ACM, 2012.

[HHDP12] Javier Hernandez, Mohammed Ehsan Hoque, Will Drevo, and Rosalind Picard, *Mood Meter: Counting smiles in the wild*, Proceedings of the International Conference on Ubiquitous Computing (Ubicomp'12), ACM, 2012.

[HTK] *Hidden Markov Model Toolkit*, http://htk.eng.cam.ac.uk.

[HWSM11] Shannon E. Holleran, Jessica Whitehead, Toni Schmader, and Matthias R. Mehl, *Talking shop and shooting the breeze: A study of workplace conversation and job disengagement among stem faculty*, Social Psychological and Personality Science (2011).

[INT08] *Intel Microprocessor Quick Reference Guide*, 2008, http://www.intel.com/pressroom/kits/quickrefyr.htm.

[JHYGP09] D.B. Jayagopi, H. Hung, Chuohao Yeo, and D. Gatica-Perez, *Modeling Dominance in Group Conversations Using Nonverbal Activity Cues*, IEEE Transactions on Audio, Speech, and Language Processing **17** (2009), no. 3, 501–513.

[KAH+12] Sokol Kosta, Andrius Aucinas, Pan Hui, Richard Mortier, and Xinwen Zhang, *Thinkair: Dynamic resource allocation and parallel execution in cloud for mobile code offloading*, Proceedings of the IEEE Conference on Computer Communications (INFOCOM '12), IEEE, 2012.

[KBC09] Marcel Kockmann, Luka Burget, and Jan "Honza" Cernocky, *Brno University of Technology System for Interspeech 2009 Emotion Challenge*, Proceedings of the Interspeech'09, 2009.

[KCHP08] Taemie Kim, Agnes Chang, Lindsey Holland, and Alex Pentland, *Meeting mediator: Enhancing group collaboration and leadership with sociometric feedback*, Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW'08), 2008.

[Kee02] Ralph L. Keeney, *Common Mistakes in Making Value Trade-Offs*, Operations Research **50** (2002), no. 6, 935–945.

[KKP99] J. M. Kahn, R. H. Katz, and K. S. J. Pister, *Next century challenges: mobile networking for "smart dust"*, Proceedings of the International Conference on Mobile Computing and Networking (MobiCom'99), ACM, 1999.

[KL10] Karthik Kumar and Yung-Hsiang Lu, *Cloud computing for mobile users: Can offloading computation save energy?*, IEEE Computer **43** (2010).

[KLGT09] Mikkel Baun Kjaergaard, Jakob Langdal, Torben Godsk, and Thomas Toftkjaer, *Entracked: energy-efficient robust position tracking for mobile devices*, Proceedings of the International Conference on Mobile Systems, Applications, and Services (MobiSys'09), ACM, 2009.

[KLJ+08] Seungwoo Kang, Jinwon Lee, Hyukjae Jang, Hyonik Lee, Youngki Lee, Souneil Park, Taiwoo Park, and Junehwa Song, *SeeMon: Scalable and Energy-efficient Context Monitoring Framework for Sensor-rich Mobile Environments*, Proceedings of the International Conference on Mobile Systems, Applications, and Services (MobiSys'08), ACM, 2008.

[KLM96] L.P. Kaelbling, M.L. Littman, and Andrew Moore, *Reinforcement Learning: A Survey*, Journal of Artificial Intelligence Research **4** (1996), 237–285.

[KLNA09] Joonas Kukkonen, Eemil Lagerspetz, Petteri Nurmi, and Mikael Andersson, *BeTelGeuse: A Platform for Gathering and Processing Situational Data*, IEEE Pervasive Computing **8** (2009), no. 2, 49–56.

[KNLZ07] A. Kansal, S. Nath, J. Liu, and F. Zhao, *Senseweb: an infrastructure for shared sensing*, IEEE MultiMedia **14** (2007), no. 4, 8–13.

[KPKB10] R. Kemp, N. Palmer, T. Kielmann, and H. Bal, *Cuckoo: a computation offloading framework for smartphones*, Proceedings of the International Conference on Mobile Computing, Applications, and Services (MobiCASE'10), 2010.

[KR76] Ralph Keeney and Howard Raiffa, *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*, John Wiley & Sons, 1976.

[Kri10] Mads D. Kristensen, *Scavenger: Transparent development of efficient cyber foraging applications*, Proceedings of the Eighth Annual IEEE International Conference on Pervasive Computing and Communications (PerCom'10), IEEE, 2010.

[KSFS05] Tim Kindberg, Mirjana Spasojevic, Rowanne Fleck, and Abigail Sellen, *The ubiquitous camera: An in-depth study of camera phone use*, IEEE Pervasive Computing **4** (2005), no. 2, 42–50.

[LBBP+11] Hong Lu, A. Bernheim Brush, Bodhi Priyantha, Amy Karlson, and Jie Liu, *SpeakerSense: Energy efficient unobtrusive speaker identification on mobile phones*, Proceedings of the International Conference on Pervasive Computing (Pervasive'11), Springer, 2011.

[LDG$^+$02] Mark Liberman, Kelly Davis, Murray Grossman, Nii Martey, and John Bell, *Emotional prosody speech and transcripts*, 2002.

[LEMC12] I. Leontiadis, C. Efstratiou, C. Mascolo, and J. Crowcroft, *Senshare: Transforming sensor networks to multi-application sensing infrastructures*, Proceedings of the European Conference on Wireless Sensor Networks (EWSN'12), 2012.

[LFO$^+$07] Joshua Lifton, Mark Feldmeier, Yasuhiro Ono, Cameron Lewis, and Joseph A. Paradiso, *A platform for ubiquitous sensor deployment in occupational and domestic environments*, Proceedings of the Sixth International Symposium on Information Processing in Sensor Networks (IPSN'07), ACM, 2007.

[LFR$^+$06] Pamela J. Ludford, Dan Frankowski, Ken Reily, Kurt Wilms, and Loren Terveen, *Because I carry my cell phone anyway: functional location-based reminder applications*, Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'06), ACM, 2006.

[LH10] Juong-Sik Lee and Baik Hoh, *Sell your experiences: A market mechanism based incentive for participatory sensing*, Proceedings of the International Conference on Pervasive Computing and Communications (Percom'10), IEEE, 2010.

[LIG] *lightblue*, http://lightblue.sourceforge.net.

[LJM$^+$12] Youngki Lee, Younghyun Ju, Chulhong Min, Seungwoo Kang, Inseok Hwang, and Junehwa Song, *CoMon: cooperative ambience monitoring platform with continuity and benefit awareness*, Proceedings of the International Conference on Mobile Systems, Applications, and Services (MobiSys'12), ACM, 2012.

[LKR04] G. Lu, B. Krishnamachari, and C.S. Raghavendra, *An adaptive energy-efficient and low-latency mac for data gathering in wireless sensor networks*, Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS'04), 2004.

[LLLZ11] Robert LiKamWa, Yunxin Liu, Nicholas D. Lane, and Lin Zhong, *Can your smartphone infer your mood?*, Proceedings of the International Workshop on Sensing Applications on Mobile Phones (PhoneSense'11), ACM, 2011.

[LLLZ13] Robert LiKamWa, Yunxin Liu, Nicholas Lane, and Lin Zhong, *MoodScope: Building a Mood Sensor from Smartphone Usage Patterns*, Proceedings of the International Conference on Mobile Systems, Applications, and Services (MobiSys'13), ACM, 2013.

[LM02] Seon-Woo Lee and K. Mase, *Activity and location recognition using wearable sensors*, Pervasive Computing, IEEE **1** (2002), no. 3, 24 – 32.

[LML$^+$10] N.D. Lane, E. Miluzzo, Hong Lu, D. Peebles, T. Choudhury, and A.T. Campbell, *A survey of mobile phone sensing*, Communications Magazine (2010).

[LOIP10] Tom Lovett, Eamonn O'Neill, James Irwin, and David Pollington, *The calendar as a sensor: analysis and improvement using data fusion with social networks and location*, Proceedings of the 12th ACM International Conference on Ubiquitous Computing (Ubicomp'10), ACM, 2010.

[LPH$^+$12] Jie Liu, Bodhi Priyantha, Ted Hart, Heitor S. Ramos, Antonio A.F. Loureiro, and Qiang Wang, *Energy efficient GPS sensing with cloud offloading*, Proceedings of the ACM Conference on Embedded Network Sensor Systems (SenSys'12), ACM, 2012.

[LPL$^+$09] Hong Lu, Wei Pan, Nicholas D. Lane, Tanzeem Choudhury, and Andrew T. Campbell, *SoundSense: Scalable sound sensing for people-centric applications on mobile phones*, Proceedings of the International Conference on Mobile Systems, Applications, and Services (MobiSys'09), ACM, 2009.

[LPR$^+$13] Neal Lathia, Veljko Pejovic, Kiran K. Rachuri, Cecilia Mascolo, Mirco Musolesi, and Peter J. Rentfrow, *Smartphones for large-scale behaviour change interventions*, IEEE Pervasive Computing, Special Issue - Understanding and Changing Behavior (2013).

[LRC$^+$12] Hong Lu, Mashfiqui Rabbi, Tanzeem Choudhury, Daniel Gatica-Perez, and Andrew Campbell, *StressSense: Detecting Stress in Unconstrained Acoustic Environments using Smartphones* , Proceedings of the International Conference on Ubiquitous Computing (UbiComp'12), ACM, 2012.

[LRMR13] Neal Lathia, Kiran K. Rachuri, Cecilia Mascolo, and Peter J. Rentfrow, *Contextual dissonance: Design bias in sensor-based experience sampling methods*, Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp'13), ACM, 2013.

[LWA$^+$08] Wu Lynn, Benjamin N Waber, Sinan Aral, Erik Brynjolfsson, and Alex Pentland, *Mining Face-to-Face Interaction Networks using Sociometric Badges: Predicting Productivity in an IT Configuration Task*, Proceedings of the International Conference on Information Systems (ICIS'08), 2008.

[LYL$^+$10] Hong Lu, Jun Yang, Zhigang Liu, Nicholas Lane, Tanzeem Choudhury, and Andrew Campbell, *The Jigsaw Continuous Sensing Engine for Mobile Phone Applications*, Proceedings of the ACM Conference on Embedded Network Sensor Systems (SenSys'10), ACM, 2010.

[MAZ$^+$11] Justin Manweiler, Sharad Agarwal, Ming Zhang, Romit Roy Choudhury, and Paramvir Bahl, *Switchboard: a matchmaking system for multiplayer mobile games*, Proceedings of the International Conference on Mobile Systems, Applications, and Services (MobiSys'11), ACM, 2011.

[MCR$^+$10] Emiliano Miluzzo, Cory T. Cornelius, Ashwin Ramaswamy, Tanzeem Choudhury, Zhigang Liu, and Andrew T. Campbell, *Darwin Phones: The Evolution of Sensing and Inference on Mobile Phones*, Proceedings of the International Conference on Mobile Systems, Applications, and Services (MobiSys'10), ACM, 2010.

[MGP06] Matthias R. Mehl, Samuel D. Gosling, and James W. Pennebaker, *Personality in Its Natural Habitat: Manifestations and Implicit Folk Theories of Personality in Daily Life*, Journal of Personality and Social Psychology **90** (2006), no. 5, 862–877.

[MHO04] Kirk Martinez, Jane K. Hart, and Royan Ong, *Environmental sensor networks*, IEEE Computer **37** (2004), 50–56.

[MKL$^+$10] E. Margaret Morris, Qusai Kathawala, K. Todd Leen, E. Ethan Gorenstein, Farzin Guilak, Michael Labhard, and William Deleeuw, *Mobile Therapy: Case Study Evaluations of a Cell Phone Application for Emotional Self-Awareness*, Journal of Medical Internet Research **12** (2010), no. 2, e10.

[MLF$^+$08] Emiliano Miluzzo, Nicholas D. Lane, Kristóf Fodor, Ronald Peterson, Hong Lu, Mirco Musolesi, Shane B. Eisenman, Xiao Zheng, and Andrew T. Campbell, *Sensing Meets Mobile Social Networks: The Design, Implementation and Evaluation of the CenceMe Application*, Proceedings of the ACM International Conference on Embedded Network Sensor Systems (SenSys'08), ACM, 2008.

[MML$^+$08] Mirco Musolesi, Emiliano Miluzzo, Nicholas D. Lane, Shane B. Eisenman, T. Choudhury, and Andrew T. Campbell, *Integrating sensor presence into virtual worlds using mobile phones*, Proceedings of the 6th ACM conference on Embedded network sensor systems (SenSys'08), ACM, 2008.

[MOT13] *MEMSIC WSN Nodes*, 2013, http://www.memsic.com/wireless-sensor-networks/.

[MP01] Matthias Mehl and James Pennebaker, *The Electronically Activated Recorder (EAR): A device for sampling naturalistic daily activities and conversations*, Behavior Research Methods **33** (2001), 517–523.

[MPF$^+$10] M. Musolesi, M. Piraccini, K. Fodor, A. Corradi, and A. T. Campbell, *Supporting Energy-Efficient Uploading Strategies for Continuous Sensing Applications on Mobile Phones*, Proceedings of the International Conference on Pervasive Computing (Pervasive'10), LCNS Springer, 2010.

[MPR08] Prashanth Mohan, Venkata N. Padmanabhan, and Ramachandran Ramjee, *Nericell: rich monitoring of road and traffic conditions using mobile smartphones*, Proceedings of the ACM Conference on Embedded Network Sensor Systems (SenSys'08), ACM, 2008.

[MRG$^+$11] Mohamed Musthag, Andrew Raij, Deepak Ganesan, Santosh Kumar, and Saul Shiffman, *Exploring micro-incentive strategies for participant compensation in high-burden studies*, Proceedings of the International Conference on Ubiquitous Computing (Ubicomp'11), ACM, 2011.

[N95] *Nokia N95*, http://www.forum.nokia.com/devices/N95.

[Nat12] Suman Nath, *Ace: exploiting correlation for energy-efficient and continuous context sensing*, Proceedings of the International Conference on Mobile Systems, Applications, and Services (MobiSys'12), ACM, 2012.

[NEP] *Nokia Energy Profiler*, http://store.ovi.com/content/73969.

[NT89] Kumpati S. Narendra and Mandayam A. L. Thathachar, *Learning Automata: An Introduction*, Prentice-Hall, Inc., 1989.

[OP10a] Daniel Olguin and Alex Pentland, *Assessing Group Performance from Collective Behavior*, Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW'10), ACM, 2010.

[OP10b] Daniel Olguin Olguin and Alex Pentland, *Sensor-based organisational design and engineering*, International Journal of Organisational Design and Engineering (2010).

[OWK+09] Daniel Olguin Olguin, Benjamin Waber, Taemie Kim, Akshay Mohan, Koji Ara, and Alex Pentland, *Sensible organizations: Technology and methodology for automatically measuring organizational behavior*, IEEE Transactions on Systems, Man, and Cybernetics **39** (2009).

[PEK+06] J. Parkka, M. Ermes, P. Korpipaa, J. Mantyjarvi, J. Peltola, and I. Korhonen, *Activity classification using realistic data from wearable sensors*, IEEE Transactions on Information Technology in Biomedicine **10** (2006), no. 1, 119 –128.

[Pen08] Alex (Sandy) Pentland, *Honest Signals: How They Shape Our World*, The MIT Press, 2008.

[PFS+09] G.P. Perrucci, F.H.P. Fitzek, G. Sasso, W. Kellerer, and J. Widmer, *On the impact of 2g and 3g network usage for mobile phones' battery life*, European Wireless Conference (EW'09), 2009.

[PH11] Unkyu Park and John Heidemann, *Data Muling with Mobile Phones for Sensornets*, Proceedings of the ACM Conference on Embedded Network Sensor Systems (SenSys'11), ACM, 2011.

[PHZ+11] Abhinav Pathak, Y. Charlie Hu, Ming Zhang, Paramvir Bahl, and Yi-Min Wang, *Fine-grained power modeling for smartphones using system call tracing*, Proceedings of the ACM European Conference on Computer Systems (EuroSys'11), ACM, 2011.

[PHZ12] Abhinav Pathak, Y. Charlie Hu, and Ming Zhang, *Where is the energy spent inside my app?: fine grained energy accounting on smartphones with eprof*, Proceedings of the ACM European Conference on Computer Systems (EuroSys'12), ACM, 2012.

[PKG10] J. Paek, J. Kim, and R. Govindan, *Energy-efficient rate-adaptive GPS-based positioning for smartphones*, Proceedings of the International Conference on Mobile Systems, Applications, and Services (MobiSys'10), ACM, 2010.

[PPC+12] Jun-Geun Park, Ami Patel, Dorothy Curtis, Jonathan Ledlie, and Seth Teller, *Online pose classification and walking speed estimation using hand-*

held devices, Proceedings of the International Conference on Ubiquitous Computing (Ubicomp'12), ACM, 2012.

[PR91] D. L. Paulhus and D. B. Reid, *Enhancement and Denial in Socially Desirable Responding*, Journal of Personality and Social Psychology **60** (1991), no. 2, 307–317.

[PSZL09] Chunyi Peng, Guobin Shen, Yongguang Zhang, and Songwu Lu, *Point&Connect: intention-based device pairing for mobile phone users*, Proceedings of the International Conference on Mobile Systems, Applications, and Services (MobiSys'09), ACM, 2009.

[PYK] *Pyke*, http://pyke.sourceforge.net/.

[PYS] *Python for S60*, https://garage.maemo.org/projects/pys60.

[QBRCN11] Chuan Qin, Xuan Bao, Romit Roy Choudhury, and Srihari Nelakuditi, *Tagsense: a smartphone-based approach to automatic image tagging*, Proceedings of the International Conference on Mobile Systems, Applications, and Services (MobiSys'11), ACM, 2011.

[Rac12] Kiran K. Rachuri, *Emotionsense: Emotion recognition and social sensing based on smart phones (demo)*, Proceedings of the 1st ACM Workshop on Mobile Systems for Computational Social Science (co-located with ACM MobiSys'12), 2012.

[RDML05] Nishkam Ravi, Nikhil Dandekar, Preetham Mysore, and Michael L. Littman, *Activity recognition from accelerometer data*, Proceedings of the 17th Conference on Innovative Applications of Artificial Intelligence - Volume 3 (IAAI'05), AAAI Press, 2005.

[REL$^+$13] Kiran K. Rachuri, Christos Efstratiou, Ilias Leontiadis, Cecilia Mascolo, and Peter Jason Rentfrow, *METIS: Exploring mobile phone sensing offloading for efficiently supporting social sensing applications*, Proceedings of the IEEE Pervasive Computing and Communication Conference (IEEE PerCom'13), IEEE, 2013.

[Rey02] D. A. Reynolds, *An overview of automatic speaker recognition technology*, Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'02), IEEE, 2002.

[RH10a] Andrew Rice and Simon Hay, *Decomposing power measurements for mobile devices*, Proceedings of the Pervasive Computing and Communication Conference (PerCom'10), IEEE, 2010.

[RH10b] ______, *Measuring mobile phone energy consumption for 802.11 wireless networking*, Pervasive and Mobile Computing **6** (2010), no. 6.

[Ric00] Golden G. Richard, *Service advertisement and discovery: enabling universal device cooperation*, IEEE Internet Computing **4** (2000), no. 5, 18–26.

[RM11] Kiran K. Rachuri and Cecilia Mascolo, *Smart phone based systems for social psychological research: Challenges and design guidelines*, Proceedings of the 3rd International Workshop on Wireless of the Students, by the Students, and for the Students (ACM S3'11), ACM, 2011.

[RMB$^+$10] Sasank Reddy, Min Mun, Jeff Burke, Deborah Estrin, Mark Hansen, and Mani Srivastava, *Using mobile phones to determine transportation modes*, ACM Transactions on Sensor Networks **6** (2010), no. 2, 1–27.

[RMM10a] Kiran K. Rachuri, Mirco Musolesi, and Cecilia Mascolo, *Energy-Accuracy Trade-offs in Querying Sensor Data for Continuous Sensing Mobile Systems*, Proceedings of the Mobile Context Awareness: Capabilities, Challenges and Applications Workshop, 2010.

[RMM$^+$10b] Kiran K. Rachuri, Mirco Musolesi, Cecilia Mascolo, Peter J. Rentfrow, Chris Longworth, and Andrius Aucinas, *EmotionSense: A Mobile Phones based Adaptive Platform for Experimental Social Psychology Research*, Proceedings of the International Conference on Ubiquitous Computing (UbiComp'10), ACM, 2010.

[RMM12] Kiran K. Rachuri, Cecilia Mascolo, and Mirco Musolesi, *Energy-accuracy trade-offs of sensor sampling in smart phone based sensing systems*, Mobile Context Awareness, Book Chapter, Springer (2012).

[RMMR11] Kiran K. Rachuri, Cecilia Mascolo, Mirco Musolesi, and Peter J. Rentfrow, *SociableSense: Exploring the Trade-offs of Adaptive Sampling and Computation Offloading for Social Sensing*, Proceedings of the International Conference on Mobile Computing and Networking (MobiCom'11), ACM, 2011.

[RNRR10] Eric Rozner, Vishnu Navda, Ram Ramjee, and Shravan Rayanchu, *NAPman: Network-Assisted Power Management for WiFi Devices*, Proceedings of the International Conference on Mobile Systems, Applications, and Services (Mobisys'10), ACM, 2010.

[ROPT05] Mika Raento, Antti Oulasvirta, Renaud Petit, and Hannu Toivonen, *ContextPhone: A Prototyping Platform for Context-Aware Mobile Applications*, IEEE Pervasive Computing **4** (2005), no. 2, 51–59.

[RPS$^+$10] Moo-Ryong Ra, Jeongyeup Paek, Abhishek B. Sharma, Ramesh Govindan, Martin H. Krieger, and Michael J. Neely, *Energy-Delay Tradeoffs in Smartphone Applications*, Proceedings of the ACM International Conference on Mobile Systems, Applications and Services (MobiSys'10), ACM, 2010.

[SFK$^+$05] A. Stolcke, L. Ferrer, S. Kajarekar, E. Shriberg, and A. Venkataraman, *MLLR transforms as features in speaker recognition*, Proceedings of the Interspeech'05, 2005.

[SG3] *Samsung Galaxy S III*, http://www.samsung.com/uk/consumer/mobile-devices/smartphones/android/GT-I9300MBDBTU-spec.

[SNR$^+$09] Ashish Sharma, Vishnu Navda, Ramachandran Ramjee, Venkata N. Padmanabhan, and Elizabeth M. Belding, *Cool-tether: energy efficient on-the-fly wifi hot-spots using mobile phones*, Proceedings of the ACM International Conference on Emerging Networking Experiments and Technologies (CoNEXT'09), ACM, 2009.

[SP12] Vijay Srinivasan and Thomas Phan, *An Accurate Two-Tier Classifier for Efficient Duty-Cycling of Smartphone Activity Recognition Systems*, Proceedings of the Third International Workshop on Sensing Applications on Mobile Phones (PhoneSense'12), ACM, 2012.

[SRL03] B. Schuller, G. Rigoll, and M. Lang, *Hidden Markov model-based speech emotion recognition*, Proceedings of the IEEE International Conference on Multimedia and Expo (ICME'03), IEEE, 2003.

[SS94] A. A. Stone and S. Shiffman, *Ecological momentary assessment (EMA) in behavioral medicine*, Annals of Behavioral Medicine (1994), 199–202.

[TKFH06] Karen P. Tang, Pedram Keyani, James Fogarty, and Jason I. Hong, *Putting people in their place: an anonymous and privacy-sensitive approach to collecting sensed data in location-based applications*, Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'06), ACM, 2006.

[TML$^+$08] David Tacconi, Oscar Mayora, Paul Lukowicz, Bert Arnrich, Cornelia Setz, Gerhard Troester, and Christian Haring, *Activity and emotion recognition to support early diagnosis of psychiatric diseases*, Proceedings of the International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth 2008), 2008.

[Tou99] Roger Tourangeau, *Remembering what happened: Memory errors and survey report*, The Science of Self-Report: Implications for research and practice (1999), 29–48.

[TRB$^+$11] Arvind Thiagarajan, Lenin Ravindranath, Hari Balakrishnan, Samuel Madden, and Lewis Girod, *Accurate, low-energy trajectory mapping for mobile devices*, Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation (NSDI'11), USENIX Association, 2011.

[TRL$^+$09] Arvind Thiagarajan, Lenin Ravindranath, Katrina LaCurts, Samuel Madden, Hari Balakrishnan, Sivan Toledo, and Jakob Eriksson, *VTrack: accurate, energy-aware road traffic delay estimation using mobile phones*, Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys'09), ACM, 2009.

[VBH03] Marc A Viredaz, Lawrence S Brakmo, and William R Hamburgen, *Energy Management on Handheld Devices*, ACM Queue **1** (2003), no. 7, 44–52.

[VBH$^+$10] C. Villalonga, M. Bauer, V. Huang, J. Bernat, and P. Barnaghi, *Modeling of sensor data and context for the real world internet*, Proceedings of the Pervasive Computing and Communication Workshops (PerCom Workshops'10), IEEE, 2010.

[VF11] M. Valla and C. Fra, *Exploiting context for automatic check-in suggestion and validation*, Proceedings of the IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops'11), IEEE, 2011.

[VRC11] Narseo Vallina-Rodriguez and Jon Crowcroft, *ErdOS: achieving energy savings in mobile OS*, Proceedings of the Sixth International Workshop on MobiArch (MobiArch'11), ACM, 2011.

[VRC13] N. Vallina-Rodriguez and J. Crowcroft, *Energy management techniques in modern mobile handsets*, IEEE Communications Surveys Tutorials **15** (2013), no. 1, 179–198.

[vRGMF09] F. von Reischach, D. Guinard, F. Michahelles, and E. Fleisch, *A mobile product recommendation system interacting with tagged products*, Proceedings of the International Conference on Pervasive Computing and Communications (PerCom'09), IEEE, 2009.

[WALW$^+$06] Geoffrey Werner-Allen, Konrad Lorincz, Matt Welsh, Omar Marcillo, Jeff Johnson, Mario Ruiz, and Jonathan Lees, *Deploying a wireless sensor network on an active volcano*, IEEE Internet Computing **10** (2006), no. 2, 18–25.

[WBCK08] Danny Wyatt, Jeff Bilmes, Tanzeem Choudhury, and James A. Kitts, *Towards the Automated Social Analysis of Situated Speech Data*, Proceedings of the International Conference on Ubiquitous Computing (UbiComp'08), ACM, 2008.

[WCC$^+$12] Tianyu Wang, Giuseppe Cardone, Antonio Corradi, Lorenzo Torresani, and Andrew T. Campbell, *WalkSafe: a pedestrian safety app for mobile phone users who walk and talk while crossing roads*, Proceedings of the 12th Workshop on Mobile Computing Systems and Applications (HotMobile'12), ACM, 2012.

[WD10] Christian Poellabauer Waltenegus Dargie, *Fundamentals of Wireless Sensor Networks: Theory and Practice*, John Wiley & Sons., 2010.

[WHFaG92] Roy Want, Andy Hopper, Veronica Falcão, and Jonathan Gibbons, *The active badge location system*, ACM Transactions on Information Systems **10** (1992), no. 1, 91–102.

[WLA$^+$09] Yi Wang, Jialiu Lin, Murali Annavaram, Quinn A. Jacobson, Jason Hong, Bhaskar Krishnamachari, and Norman Sadeh, *A Framework of Energy Efficient Mobile Sensing for Automatic User State Recognition*, Proceedings of the International Conference on Mobile Systems, Applications, and Services (MobiSys'09), ACM, 2009.

[WOKP10] Benjamin Waber, Daniel Olguin Olguin, Taemie Kim, and Alex Pentland, *Productivity through Coffee Breaks: Changing Social Networks by Changing Break Structure*, Proceedings of the 30th International Sunbelt Social Network Conference (SSNC'10), 2010.

[XRC$^+$04] Ning Xu, Sumit Rangwala, Krishna Kant Chintalapudi, Deepak Ganesan, Alan Broad, Ramesh Govindan, and Deborah Estrin, *A wireless sensor net-*

work for structural monitoring, Proceedings of the International Conference on Embedded Networked Sensor Systems (SenSys'04), ACM, 2004.

[XSK$^+$10] Yu Xiao, Petri Savolainen, Arto Karppanen, Matti Siekkinen, and Antti Ylä-Jääski, *Practical power modeling of data transmission over 802.11g for wireless applications*, Proceedings of the First International Conference on Energy-Efficient Computing and Networking (e-Energy'10), ACM, 2010.

[YAH] *Number of smartphones around the world top 1 billion - projected to double by 2015, Yahoo Finance, 2012*, http://finance.yahoo.com/news/number-smartphones-around-world-top-122000896.html.

[Yan06] Guang-Zhong Yang, *Body Sensor Networks*, Springer, 2006.

[YCG$^+$12] Tingxin Yan, David Chu, Deepak Ganesan, Aman Kansal, and Jie Liu, *Fast app launching for mobile devices using predictive user context*, Proceedings of the International Conference on Mobile Systems, Applications, and Services (MobiSys'12), ACM, 2012.

[YKE$^+$10] Gokhan Yavuz, Mustafa Kocak, Gokberk Ergun, Hande Ozgur Alemdar, Hulya Yalcin, Ozlem Durmaz Incel, and Cem Ersoy, *A smartphone based fall detector with online location support*, Proceedings of the International Workshop on Sensing Applications on Mobile Phones (PhoneSense'10), ACM, 2010.

[ZAT$^+$05] Alex J. Zautra, Glenn G. Affleck, Howard Tennen, John W. Reich, and Mary C. Davis, *Dynamic Approaches to Emotions and Stress in Everyday Life: Bolger and Zuckerman Reloaded with Positive as Well as Negative Affects*, Journal of Personality **73** (2005), 1511–1538.

[ZCCM12] Zengbin Zhang, David Chu, Xiaomeng Chen, and Thomas Moscibroda, *Swordfight: Enabling a new class of phone-to-phone action games on commodity phones*, Proceedings of the International Conference on Mobile Systems, Applications, and Services (MobiSys'12), ACM, 2012.

[ZCCT12] Andong Zhan, Marcus Chang, Yin Chen, and Andreas Terzis, *Accurate caloric expenditure of bicyclists using cellphones*, Proceedings of the ACM Conference on Embedded Network Sensor Systems (SenSys'12), ACM, 2012.

[ZLL$^+$11] Gang Zhou, Qiang Li, Jingyuan Li, Yafeng Wu, Shan Lin, Jian Lu, Chieh-Yih Wan, Mark D. Yarvis, and John A. Stankovic, *Adaptive and radio-*

agnostic qos for body sensor networks, ACM Transactions on Embedded Computing Systems (TECS) **10** (2011), no. 4, 48:1–48:34.

[ZMN05] F. Zhu, M.W. Mutka, and L.M. Ni, *Service discovery in pervasive computing environments*, IEEE Pervasive Computing **4** (2005), no. 4, 81–90.

[ZSLM04] Pei Zhang, Christopher M. Sadler, Stephen A. Lyon, and Margaret Martonosi, *Hardware design experiences in zebranet*, Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys'04), ACM, 2004.

[ZTQ$^+$10] Lide Zhang, Birjodh Tiwana, Zhiyun Qian, Zhaoguang Wang, Robert P. Dick, Zhuoqing Morley Mao, and Lei Yang, *Accurate online power estimation and automatic battery behavior based power model generation for smartphones*, Proceedings of the Eighth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES/ISSS'10), ACM, 2010.

[ZZL12] Pengfei Zhou, Yuanqing Zheng, and Mo Li, *How long to wait?: predicting bus arrival time with mobile phone based participatory sensing*, Proceedings of the International Conference on Mobile Systems, Applications, and Services (MobiSys'12), ACM, 2012.