Scalable and Energy EfficientSoftware Architecture for Human Behavioural Measurement

A Thesis Submitted to the

College of Graduate and Postdoctoral Studies

in Partial Fulfillment of the Requirements

for the degree of Master of Science

in the Department of Computer Science

University of Saskatchewan

Saskatoon

By

Osagie Osemwegie

©Osagie Osemwegie, July 2017. All rights reserved.

Permission to Use

In presenting this thesis in partial fulfilment of the requirements for a Postgraduate degree from the University of Saskatchewan, I agree that the Libraries of this University may make it freely available for inspection. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my thesis work or, in their absence, by the Head of the Department or the Dean of the College in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Saskatchewan in any scholarly use which may be made of any material in my thesis.

Requests for permission to copy or to make other use of material in this thesis in whole or part should be addressed to:

Head of the Department of Computer Science

176 Thorvaldson Building

110 Science Place

University of Saskatchewan

Saskatoon, Saskatchewan

Canada

S7N 5C9

Abstract

Understanding human behavior is central to many professions including engineering, health and the social sciences, and has typically been measured through surveys, direct observation and interviews. However, these methods are known to have drawbacks, including bias, problems with recall accuracy, and low temporal fidelity. Modern mobile phones have a variety of sensors that can be used to find activity patterns and infer the underlying human behaviors, placing a heavy load on the phone's battery. Social science researchers hoping to leverage this new technology must carefully balance the fidelity of the data with the cost in phone performance. Crucially, many of the data collected are of limited utility because they are redundant or unnecessary for a particular study question. Previous researchers have attempted to address this problem by modifying the measurement schedule based on sensed context, but a complete solution remains elusive. In the approach described here, measurement is made contingent on sensed context and measurement objectives through extensions to a configuration language, allowing significant improvement to flexibility and reliability. Empirical studies indicate a significant improvement in energy efficiency with acceptable losses in data fidelity.

Acknowledgements

I will like to to thank my supervisor, Kevin Stanley, allowed me to conduct this work for 2 years providing me with enormous number of guidance and mentorship. I would like to thank T. Paul, W. Qian and A. Mann for use of their entropy rate estimation code, and the Natural Sciences and Engineering Research Council of Canada, the Saskatchewan Health Research Fund and the University of Saskatchewan for funding this work.

Contents

List of Tables

List of Figures

List of Abbreviations

SHED	**S**askatchewan **H**uman **E**thology **D**ataset
CMQ	**C**ontext **M**onitoring **Q**uery
SQL	**S**tructured **Q**uery **L**anguage
GSM	**G**lobal **S**ystem for **M**obile communications
GPS	**G**lobal **P**ositioning system
RAM	**R**andom **A**ccess **M**emory
CPU	**C**entral **P**rocessing **U**nit
RSS	**R**eceived **S**ignal **S**trength
RAPS	**R**ate **A**daptive **P**ositioning System
CAPS	**C**ell-ID **A**ided **P**ositioning **S**ystem
SEDDACCO	**SE**lf **D**ocumenting **D**ata **A**nalysis and **CO**llection **C**ode
EEG	**E**lectro**E**ncephalo**G**ram
XML	**E**xtensible **M**arkup **L**anguage
URL	**U**niform **R**esource **L**ocator
API	**A**pplication **P**rogramming **I**nterface
NALP	**N**eighbourhood **A**ctive **L**iving **P**otential

Chapter 1

Introduction

Understanding of human behaviour underpins several research fields. From emergency evacuation modelling, to understanding the spread of disease, to city planning, the dynamics of human behaviour shapes our safety, comfort and health. Classic studies in human behaviour have employed surveys, diaries and interviews [13]. While providing insight into the psychology of behaviour, these tools are limited in the information that they can acquire as it shifts the burden of data collection to the participants and participants' answers and completion rates can be limited by the participant's memory or emotional state such as if the participant having a sad day prior to completing the survey or interview [44]. Human memory and perception is often biased, causing participants and researchers alike to emphasize some events at the expense of others. Participants are also likely to avoid reporting embarrassing behaviour, or provide distorted answers to match social norms or please the researcher [44]. Participant self-reporting is subjective; designing surveys to consistently quantify behaviour across participants becomes difficult.

With recent advances in portable technologies, scientists have exploited the use of these portable devices to help reduce observation and bias encountered by researchers studying human behaviour. Devices such as GPS and step counters play a part in improving data quality as they can provide a high quality and fine-grained channel for acquiring data, simplifying the comparison between evidence gathered in behavioural research because the data collected is physically less subject to participant biases.

Purpose-built devices have been proposed to monitor individuals, areas, and environments using special purpose built hardware known as wireless sensor networks. These technologies have mainly been used in non-human monitoring applications such as observing moving animals [59], or environment like earthquakes [60]. These devices have been used for various research projects involving humans [57, 58, 64]. However, when used for human applications, they can be easily damaged during a participant's daily activities and are vulnerable to static electricity [18]. Because these devices have little value outside the purpose of the study, participants have little motivation to carry such devices. These devices are also more difficult to reconfigure, update and deploy; creating an entry barrier for potential participants reducing their overall utility.

While purpose-built devices provide some advantage over traditional tools, they suffer from significant drawbacks. However, today's smartphones provide a rich sensing platform that has produced several applications ranging from simple step counting [38] to fall detection systems [2], and general behavioural monitoring tools [1, 13, 17, 36, 55]. These tools allow the simultaneous monitoring of activity level, interpersonal contact, location and geographic context, providing a rich stream of potential data for social science researchers and practitioners. Smartphones reduce some of the limitations encountered in surveys and interviews and the smartphone is capable of collecting more data than participants' are likely to remember, such as the places they visited over the past month.

Unfortunately, the constant use of built-in smartphone sensors is challenging for resource constrained hardware platforms. Continuously capturing user context through sensors can place a heavy load on the smartphone processor, memory and sensors. In particular, the drain on energy reserves can be limiting, with utilization of all available sensors completely depleting the phone's batteries in a matter of hours [45], [42]. To address this issue, researchers have developed duty cycling strategies where the phone periodically measures and then sleeps to conserve energy, or have created hierarchical systems where low-power sensors preferentially perform measurements [9]. Simply measuring on a fixed duty cycle could be wasteful; however, depending on the research questions underlying the experiment. For example, researchers interested in shopping patterns would want high fidelity information while shopping behaviour was likely, for example while participants were in a mall, but be willing to have lower fidelity information about the rest of the participants' behaviours.

The central idea of duty cycle is to wake up device to collect sensor data periodically for a short period of time called burst length instead of continuous data collection. Figure 1.1 shows an example of a duty cycle measurement that collects data every 30 seconds for 2 minutes. The duty cycle is controlled by a scheduling process.

Choosing the most energy efficient duty cycle while still acquiring sufficiently high fidelity data to answer the behavioural research questions is a challenging problem [22], in part because the definition of "sufficiently

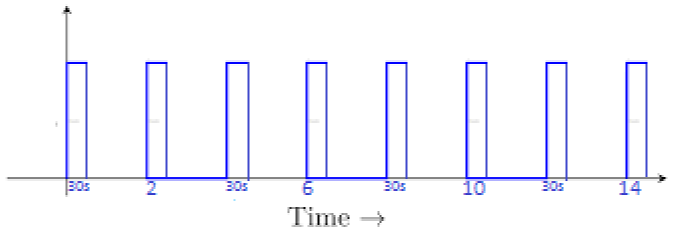


Figure 1.1: An example of a duty cycle measurement.

high fidelity" changes from experiment to experiment, and the manifestation of behaviours varies with participant, physical context, and time. Architectures which support dynamic scheduling of sensing tasks based on both a priori experimenter definition, and sensed context would be required to provide the kind of context aware duty cycle management required to negotiate an appropriate trade-off between measurement benefit and energy cost. The purpose of this thesis is to explore the effect of duty cycle scheduling on battery life and quality of data.

The problem choosing sampling frequency versus energy consumption can be explained with the following example. Consider a study that collects mobility data from users. To get close to a full day of battery life for an average smartphone, one needs about 8 minute duty cycle and a burst length of 30 seconds [7]. Collecting data for 30 seconds every 8 minutes means a portion of a user's trajectories will be missed. However, continuous collection of data makes the phone last for just few hours with current phone technologies. Continuous sensing also collects redundant data for example if a user stays at home the entire day. The goal of this work is to demonstrate that it is possible to develop a software architecture that reduces battery consumption in smartphones' data collection by supporting multiple adaptive duty cycling scheme.

In this thesis, I describe a scheduling system, built by extending the SEDDACCO measurement configuration language [26] and the iEpi measurement platform [27], which allows experimenters to specify measurement regimes and the triggering sensed context. I demonstrate the efficacy of this new system over three one week experiments, each employing different scheduling logic. I implement a form of adaptive scheduling where the measurement frequency is adjusted based on the sensed context by extending the iEpi architecture, and allow the run-time specification of schedules and heuristics through extensions to the SEDDACCO language. I show a substantial decrease in energy utilization while maintaining an acceptable data fidelity. Efficacy of the system is demonstrated through analysis of battery logs, sensor data collected, and by computing the information entropy rate of adaptively sampled and non-adaptively sampled conditions. The scheduling system was also used in two social science studies. The first study involved monitoring the food consumption habits of 17 families and the second study, known as SHED6, involved the collection of data from 74 students for a period of 30 days. The versatility of the system helped in reducing battery consumption and also allowed researchers to deploy different context-aware scheduling configurations. Though the analyses of the social science experiments are not present in this thesis, I provide some evidence to show the desirability and utility of the system.

The remainder of this thesis is organized as follows. Chapter 2 gives a relevant prior research. Chapter 3 provides the purpose and intention of the proposed framework design. Chapter 4 shows the experimental setup to prove the efficacy of the architecture. Chapter 5 shows the result from the experiment. Finally, chapter 6 shows the conclusion and future work.

Chapter 2
Related Work

The work of this thesis concerns the design and development of a flexible architecture for data collection. There is significant work on this subject ranging from the design to the issues encountered. As the focus of this thesis is not the data derived by these systems but rather the systems themselves, this review will largely address the technical aspects of related work as opposed to the behavioural monitoring implications they provide.

2.1 Sensor network programming

Engineers have created wireless sensor networks applications for areas including health care, utilities, and remote monitoring. These applications provide valuable insight into human behaviour and other health-related phenomena. However, when the behavior of the devices need to be changed, the devices traditionally have to be taken back to the developers to reprogram. This makes the system difficult to maintain and manage. In addition to this issue, researchers are forced to understand the underlying hardware and the low level coding constructs [33].

To give researchers an easier way to program wireless sensors, software developers have created various macro-programming languages by extending existing widely-used general purpose programming languages. Kairos [16], an extension to the Python language, provides a set of extensions that allows programmers to express the global behavior of a wireless sensor network by exposing simple primitive functions - read/write variables, looping through nodes, and node addressing. Kothari *et al.* [28] extended the C programming language in a language called Pleiades. Pleiades uses a central program to configure wireless sensor networks. Pleiades allows users to specify the global behavior of their programs. Pleiades' language extensions include concurrency primitives, node manipulation primitives, keywords to describe whether or not variables must be synchronized, and sensor abstractions. EcoCast [56] is a macro-programming system for wireless sensor networks. The system works as an extension to the Python programming language. It extends Python's map, reduce and filter functions to provide versions that work in a macro-programming environment, allowing for simpler processing of distributed collections of nodes.

Another approach used by engineers is to emulate frameworks commonly used by other popular programming languages so as to reduce the entry barrier. TAG [32] uses SQL-like queries to request data from a sensor network. TAG allows users to express simple, declarative queries and have them distributed and executed efficiently in networks of low-power, wireless sensors. MacroLab [19] uses a Matlab syntax to perform operation on wireless sensor data. The user writes a single program for the entire network using Matlab-like operations such as addition, find, and max. The framework executes these operations across the network in a distributed fashion or centralized fashion – whichever is most efficient for the target deployment.

Scientists have also used rule based macro-programming on sensor networks. Instead of querying raw sensor data, this system interprets the data and returns the result to the user. Two particular instances of such techniques are described in the remainder of this section.

Semantic Streams [62] allows nontechnical users to pose queries over semantic interpretations of sensor data, such as querying to know the ratio of cars to trucks in a parking garage without actually writing code to infer any of the objects from the raw sensor data. The system contains two fundamental elements: event streams and inference units. Event streams represent a flow of asynchronous events that represents a real-world event, such as an object, person or car detection and has properties such as the time or location it was detected, its speed, direction, and/or identity. Inference units are processes that operate on event streams. They infer semantic information about the world from incoming events and either generate new event streams or add the information to existing events as new properties.

FACTS [56] is a proof-of-concept middleware framework based on rules, facts, and functional abstractions. It was developed to reduce the difficulty programmers face when programming event-based systems, e.g., alerting the security guard when there are too many people in a room. The rules in FACTS express the reaction to an event, the facts represents data and functions enable interaction between FACTS and software outside of FACTS.

To the best of my knowledge, this thesis is the only work conducted on using a plug and play architecture with macro-programming. The plug and play allows modules and functionality to be added at runtime without developer adding any code to the original source code or having developers redeploy the entire system.

2.2 Management and collection of data

Smartphones come with various sensors and each sensor consumes different amounts of energy, [45] so one might need to use different duty cycles on different sensors. The data needs to be processed or stored in the phone or sent to a server. Due to such complexity, several scientists have designed dedicated software architectures for data collection.

Min [34] implemented a software architecture which is composed of four components: health monitor, a privacy controller, a sensor broker and a resource efficient sensor controller. The app known as Healthopia was developed to collect health-behaviour related data from smartphones. The health monitor is designed to continuously collect health-behaviour information. It processes sensor data and infers relevant information from the data. The privacy controller allows users to specify apps that the user would like to allow to access the the information generated by Healthopia. Sensor broker is designed to manage external health sensors like wearable devices. It sends a broadcast message to discover nearby sensors and interprets the data received from the external device. The resource efficient sensor controller allows for efficient monitoring of changes in health information. It selects the essential sensors for monitoring the queries and resolves the conflicts of the applications for shared resources.

Kang [21] proposed a middle-tier architecture called Seemon. It uses a context monitoring query (CMQ) with which users can specify various range of context. CMQ requires 3 conditions including context, alarm and duration. The context is a predefined user context programmed in Seemon like a specific location of the user e.g. at university. The alarm determines when the event is triggered and the duration determines how long the query should run. Below is the structure of the query.

```
Context <context element>

(AND <context element> )
```

```
ALARM <type>

DURATION <duration>
```

Seemon consists of four components: the context monitoring query processor, the sensor manager, the application broker, and the sensor broker. Seemon process are in three phases: query registration, query processing, and sensor control. Firstly, the CMQ is registered to the CMQ processor. The processor evaluates the queries registered over the data received from the sensor broker. Finally, the sensor manager stops unused sensors from transmitting data.

The application broker provides an interface for other applications to communicate with Seemon. It also controls privacy and security by checking if the application requesting context information is registered in an access control list [47].

Jigsaw [9] presents a pipeline architecture to facilitate efficient data collection. Jigsaw consists of three classification pipelines: accelerometer, microphone and GPS pipelines which are responsible for collecting and processing information from the individual sensor. Each sensor pipelines chooses its sampling scheme and provides contextual information from the data.

Kirovski *et al.* [23] proposed Health-OS, a platform that specifies a robust architecture for a system that unifies health-related apps. Health-OS is a middleware platform consisting of a sensing module, a mobile personal data hub and a trusted personal computer. The sensing module provides Health-OS access to the external world. Each sensing module contains: a sensor; a controller for data modeling, compression, and analysis; a communications module with encryption and error correction codebooks; memory management and an energy source. Each module, after collecting data, sends the data to the personal data hub. The data hub is responsible for storing and analyzing the data collected from the sensors. The data hub is also responsible sending data via a network connection to a server. To reduce energy consumption, the data hub would synchronize its contents relatively infrequently with the computer.

EpiCollect [1] is an Android based application which allows data like photos or GPS records to be entered and stored on a mobile phone and sent to a central web database. The web server provides analysis and visualization tools that can be used on the data by researchers. EpiCollect was designed to allow researchers to collect data points in the field to record and then automatically process data. The application consists of a series of interfaces on the smartphone to collect user input. The most recent version [61] makes use of an online builder to specify the data fields of interest; the system then uses the information to generate interfaces on the device and a database schema for storing the data. EpiCollect solves a major issue encountered by ecologists. Traditional tools often require the manual collection of data points from outdoor environments and insertion of those data into a database, processes which are tedious and prone to transcription errors [1].

2.3 Battery issues with smartphone sensors

Smartphones are powered from batteries that are limited in capacity and easily drained. This implies that managing energy is paramount for such devices. Studies based on smartphone data collection require a good understanding of energy management – how much of the energy is used by which parts of the system and under what circumstances.

Aaron *et al.* [7] shows the distribution of power consumption among different components of a modern smartphone. First, they measured the energy consumption of the different components at a circuit level using a data acquisition system and a host computer. Their result shows the RAM to consume the lowest amount of energy and the CPU, mobile radio and graphics consume the most energy. In their study, they noticed energy consumption by the GPS module to be largely independent of the signal strength and number of satellites involved, which is contrary to Naik [37]. The Bluetooth module was also noted to consume a negligible amount of energy when the CPU is not activated. From their result, one can assume that the individual sensors consume low amounts of energy, and instead, energy consumption is driven by the CPU.

Priyantha *et al.*[45] studied the energy consumption of just the sensors on smartphones and proposed the use of low-power micro controllers for data acquisition. The study showed that keeping the main CPU in the active state rather than idle state is one of the major battery drains. The study, using an HTC Touch PRO, showed that when the accelerometer is sampled, the overall power consumption increases by 370 mW even though the datasheet shows that the device consumes less than 1 mW when active. They noted that the increase comes from the need to keep the phone's main processor and associated high-power components active to access the sensor results, yielding a total power consumption that is orders of magnitude larger than the power required for sensing. They also showed that there is a high energy drain when a phone changes from a sleep state to the idle state and from the idle state to this sleep state. Their experiment showed that the phone takes approximately 900 ms to move into a sleep state and about 270 ms to exit from the sleep state.

Liu *et al.* [31] looked at battery consumption from flawed design implementations by developers. They found two common types: sensor listener misusage and sensory data underutilization. Sensor listener misuage is when a developer fails to stop a sensor from collecting data after the necessary data has been collected. Failing to unregister causes significant additional battery consumption. Sensory data underutilization is when data are collected but are not used by the application requesting it. They then propose an approach for systematic diagnosis of energy inefficiency problems in Android applications.

Continuous sensing applications are power hungry. To address energy consumption with respect to human behavior monitoring, some scientists have used a duty-cycling scheme [13, 27], specifying when and how long a sensor is activated. Having flexibility to specify a duty-cycle regime for each sensor has been shown to help conserve energy [27], but the main issue is choosing the most energy efficient duty cycle that will also guarantee quality data. Several authors have tried various monitoring schemes to reduce consumption by activating a high-power sensor only if the system is unable to detect the context of the user using a low-power sensor. Other researchers have also proposed prediction-based schemes to find regularities in mobility patterns in order to re-use data without activating sensors when a user follows previously observed patterns [3, 8, 50].

2.4 Accuracy constrained scheduling

Lin *et al.* [30] designed a prototype framework called A-LOC which automatically adapts sensed location energy and accuracy based on dynamically varying the smartphone sensor used for location. A-LOC is able to be use other sensors like Bluetooth, cell-tower and WiFi to determine location. The framework consists of four major blocks. The Dynamic Accuracy Requirement block provides the accuracy needed by the applications. The Sensor Energy Model block models the energy used by each available location sensor for obtaining location. The Dynamic Sensor Accuracy Model block models the quality of location information received using a Bayesian technique. The Sensor Selection Algorithm block determines the location sensor to be used at each time step.

RAPS [40] is a smartphone application like A-LOC, but it uses other sensors rather than GPS when results are less accurate. In contrast to A-LOC, RAPS uses a duty-cycled accelerometer to efficiently estimate user movement: whether a user is moving or not. RAPS stores the space-time history of user movements to estimate when to activate GPS by using the velocity of the device. RAPS is also able to detect GPS unavailability and subsequently turn the GPS on or off. RAPS is able to detect GPS unavailability by employing cell-tower RSS blacklisting. Finally, RAPS also uses Bluetooth to reduce uncertainty by communicating with nearby devices that uses RAPS to determine which of the devices has an higher accuracy, and subsequently uses the result with the highest accuracy.

EnTracked [24] was proposed for position tracking and trajectory tracking. EnTracked uses three strategies to determine position and trajectory: a head-aware strategy that uses compass as a turning point sensor, a distance-aware strategy that dynamically predicts how long the GPS can remain idle between successive measurement and a movement-aware strategy which uses the accelerometer to detect movement. Entracked selects the optimal strategy given the current requirements and how much power is needed.

Abdesslem *et al.* [4] propose the use of less power hungry sensors more often and enabling the more expensive sensors less frequently. They implemented this approach in an app called Senseless. Senseless is designed to make use of the accelerometer to sense movement and only uses the GPS after movements have been detected.

2.5 Prediction based scheduling

Paek *et al.* [41] proposes CAPS. CAPS uses cell-ID sequences to efficently locate a user's position with reasonable accuracy. CAPS stores the cell-ID and trajectory of paths followed by a user. It adopts a sequence matching technique to efficently determine a user's position by identifying a cell-ID sequence in the user's history. The strategy enables CAPS to exploits the fact that people take similar routes on a daily basis and have consistent cell-IDs.

Becker *et al.*. [3] explores the use of cellular handoff patterns to identify users' trajectory through a city.

Data was collected from 15 driving and train routes with roughly 20,000 residents. The authors developed two methods for classifying handoff patterns on these routes. The first method uses nearest-neighbour classification to identify routes and the second uses signal-strength. The model developed was then used to predict individual trajectories.

FreeTrack [8] measures the position of users by taking advantage of their mobility patterns, cell tower state and battery state. The system activates the high power sensors, WiFi or GPS, when the position cannot be acquired through other means such as cell connection and battery state. FreeTrack uses no pre-trained data set or centralized server. It uses WiFi fingerprints to recognize places at room-level accuracy, which the author defines as an indoor space where a user spends a certain period of time. With no historical data, FreeTrack collects data from WiFi at regular intervals to determine location. When a user stays at a place for certain period of time, a place signature is derived composed, of the location, cell connection, WiFi fingerprint, battery state, arrival time, and dwell time. If the place has not previously been observed (by comparing WiFi previously of scanned places), the system creates a new place. The system minimizes the use of the high power sensors if the place has been re-visited by user. FreeTrack contains a location inference engine that models the user mobility pattern using a Hidden Markov Model, which decides when to trigger a high power sensor.

Thiagarajan *et al.* presented VTrack [54] a traffic monitoring system that uses a Hidden Markov Model (HMM) to model a vehicle trajectory over an area. VTrack makes use of GPS and WiFi for location and performs map matching over the data and produces travel time estimates for each travel path. VTrack provides real-time estimates to user. It also compiles a database of historic travel delays on road segments. They evaluated their study using location data of about 800 hours of actual commuter drivers. The data was gathered from an iPhone and embedded in-car GPS and WiFi radios.

This section shows the battery issue and the approach that has been used in solving the issue. However, each approach only target a single use case, location. There is yet a single solution to accommodate different approach into a single solution where users can choose. My approach uses an extensible architecture where different monitoring approach can be used.

2.6 Entropy rate

Entropy rate has been used for measuring the degree of predictability characterizing a time series. Entropy rate or the smallest amount of information required to represent a message, was used by Song [51] to determine the predictability of individuals around a major urban center. Their analysis demonstrated a modest entropy rate which was independent of the typical range of a user's trajectory. Their result showed that a user's mobility has a potential to be 90 percent predictable. However, it is unclear whether this entropy rate was due to their population or the data collection approach.

Gavin [50] shows that the approach to deriving the upper bound of predictability in human mobility provided by Song [51] can be significantly tightened by integrating topological constraints, providing a far lower and hence more accurate upper bound. Their result indicates that a lower predictability of 70 percent.

Qian *et al.* [46] analyze a smaller but high-resolution data set to determine the impact of spatial data encoding, resolution, and filtering on the mobility entropy of the study population. Their data set includes GPS and WiFi proximity traces from 38 participants. Their results showed that higher resolution mobility patterns are less predictable than coarser ones. Motivated in part by the fact that different spatial-temporal granularity yields different entropy rates as noted by Qian *et al.* [46], Osgood *et al.* [39] showed a methodology that estimates the differences of mobility entropy at different spatial and temporal scales.

This paper uses entropy rate as a metric determine the information lost in different adaptive measuring scheme.

2.7 iEpi previous work

The system developed in this thesis is the third iteration of a larger research project named iEpi, which has been in development for several years. iEpi was was motivated by an earlier system called Flunet [18]. Flunet used MicaZ motes in a wireless sensor network to capture human contact data throughout the 2009-2010 pandemic flu season. Flunet suffered from several issues. The motes were easily damaged, forgotten and are difficult to program. These issues led to the first iteration of iEpi.

iEpi was further motivated by challenges in understanding the spread of infectious disease [44]. Scientists have long used mathematical and computational models to study the spread of diseases. However, models can become inaccurate without fine-grained data. The first iteration of iEpi attempted to solve this issue by providing fine-grained behavioural data to improve the quality of influenza-like disease modelling. The first use of iEpi was to collect data over five weeks from 39 participants [17]. While significant battery problems were encountered, 45 million data records were collected.

The first iteration of the iEpi architecture used a pipeline architecture composed of stream, task, data logger, upload and database. The Stream collects sensor data and the task controls the stream's schedule. After data is collected, the data logger is invoked to store the data, which is later retrieved for upload to a server [17]. At this point in its development, iEpi used an extremely restricted version of SEDDACCO for its configuration [26]. It used regular expressions to extract tasks to be performed.

The second iteration of iEpi focused on modularity and extensibility. SEDDACCO was redesigned to use a more robust Antlr library for parsing and compiling the configuration file. Other functionality was added SEDDACCO, like conditions and task-chaining. A detail overview of the architecture can be found in 3.4.

This thesis introduces the third iteration of iEpi. This iteration aims to increase the flexibility of iEpi and reduce battery consumption. iEpi uses a plug-and-play architecture for modularity and flexibility. The plug and play architecture allows researchers to add different scheduling algorithms that specifically target their study interest, allowing for reduced battery consumption, as it records only information deemed useful. The behavior of individual modules and scheduling algorithms are configured using the SEDDACCO programming language. This allows researchers or scientists to reconfigure the devices at runtime without redeployment.

Chapter 3

iEpi Version 3

iEpi is an end-to-end system for collecting and analyzing contextual data using a combination of smartphone and server technology. Data is collected and encrypted via smartphone and then uploaded to a server. iEpi is capable of recording data from on-board sensors as well as external devices such as smartwatches or fitness watches.

3.1 Logger

The logger is an Android-based application that collects sensor data. The logger uses a pipe and filter software architecture to organize different processes and tasks. The foundation of the system is a series of Android threads called processes. A process, not to be confused with an Android process, controls the execution of a pipeline of tasks, known as a task chain. A task is the unit that does work like data collection or data processing. All task output is encrypted and saved on the device until it is uploaded to the server or manually removed. Each processes has a scheduler that controls when a process starts and stops. The schedule for each processes is configured in a configuration language called SEDDACCO, discussed in section 3.3. The logger recording capabilities are discussed below.

3.1.1 Recording capabilities

- Proximity: iEpi is able to collect Bluetooth data. Bluetooth data allows the device to detect nearby devices and their type, e.g. smartphone, printer or computer. As smartphone users typically carry their smartphone, the data could be used to detect people who are close by.

- Location: They are several methods from which location can be estimated. iEpi is able to collect data from the following smartphone components:

  - GPS data is collected by iEpi with which it is able to detect speed and location. GPS is known to provide the most accurate result for location, with accuracy as high as within 5 meter when outodors.
  - Celltower proximity is able to give a coarse representation of the user's location. A slightly more accurate result can be obtained from celltower by triangulation if three or more celltowers are in range.
  - WiFi, like celltower is able to provide the device location if the access point's location is known. Trilateration can also be used if more than three routers are in proximity.
  - Bluetooth data can be used to find location if the device is in proximity to a known stationary device, such as a printer.

- Activity and Motion: iEpi is able to collect sensor data from gyroscope, accelerometer and magnetometer. For physical activity recognition, the data from sensors, such as an accelerometer and a gyroscope, have been utilized in research studies. A combination of a gyroscope, a magnetometer and an accelerometer can be used to infer the orientation of the device, a useful feature for other context sensing algorithms.

- Battery Status: iEpi is able to record the energy level, battery temperature, voltage and the state (charging or discharging) of the battery, including the source it is charging from (wall socket or USB). One can use this data to model power consumption or to infer if the phone is with the participant or not. This data has also be used to determine compliance of participants (see 5.1.1). As iEpi can also collect currently running applications, one is able to infer which applications takes the most battery.

- Survey: iEpi is able to launch surveys. The survey can be context triggered, triggered by participants or triggered based on a schedule. How the survey launches is configured using SEDDACCO (see 3.3). The survey can be configured at runtime on the server. iEpi frequently checks if there is a new survey file to be downloaded.

- External Devices: iEpi can be configured to receive data from external sensor devices. Below are some devices that iEpi have been used to collect data from
  - Wii Board,
  - EEG,
  - Heart rate monitor, and
  - Skin Temperature monitor.

3.2 Autoparser

When data are collected by the logger, it is compressed, encrypted and sent to a backend server. The autoparser is a tool written in Scala that is used to decrypt, process and store the data in a database.

The autoparser is configured through the use of an XML file. The XML file contains database configuration and details needed to decrypt the file. The authoparser performs the following steps to parse the incoming data.

- Configuration file validation. The configuration is validated to make sure the XML is in the proper format and that all of the information required to parse the data is present in the file.
- Database connection. The database login details which includes the URL, username and password.
- Database creation. The database tables are created if the user specifies, otherwise the system uses the already created table in the database. If no table is created and the user did not specify table creation, the autoparser throws an exception.
- Parsing. The files are decompressed, decrypted and stored in the database.

3.3 SEDDACCO

SEDDACCO (SElf Documenting Data Analysis and Collection COde) [26] is a custom configuration language for iEpi which takes an English-like, declarative-based syntax which allow the programmer to specify the configuration of the system rather than the implementation. The standard task syntax is given below.

- Program Structure:

```
before tasks start:
    [Startup Block]
done;
[Program Body].
```

- Startup Block:

```
<task clause> [condition clause] [parameter clause]
[chained tasks clause] [semicolon].
```

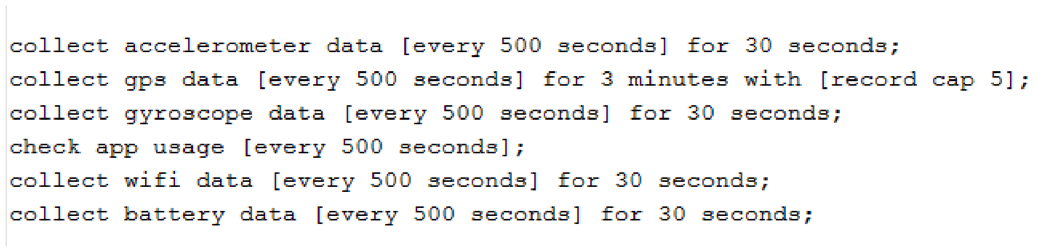
- Program Body:

```
<task clause> <schedule clause> [condition clause]
[parameter clause] [chained tasks clause] [semicolon].
```

- Chained Tasks Clause:

```
<task clause> [parameter clause]
[chained tasks clause][semicolon].
```

The syntax changes slightly depending on the context for practical reasons. For instance, it makes little sense to provide a schedule to a task that will be executed exactly once at the beginning of the program. Figure 3.1 shows an example of how SEDDACCO is used in an actual study [26].



```
collect accelerometer data [every 500 seconds] for 30 seconds;
collect gps data [every 500 seconds] for 3 minutes with [record cap 5];
collect gyroscope data [every 500 seconds] for 30 seconds;
check app usage [every 500 seconds];
collect wifi data [every 500 seconds] for 30 seconds;
collect battery data [every 500 seconds] for 30 seconds;
```

Figure 3.1: Sample of SEDDACCO language

3.4 Previous iEpi software architecture

The basic functionality described in this section is primary the work of Knowles [25]. I presented an overview of the architecture and its limitations [27]. In particular, the relevant components required to address duty-cycle inefficiency - the Process, Scheduler, Task and Condition components - are described.

iEpi consists of series of parallel processes. Each process controls the execution of a chain of tasks. Each task contains atomic actions of gathering, manipulating or storing data from sensor streams. To reduce battery consumption, SEDDACCO is used to control when a process starts and for how long the process should run. Before each task is started, a user specified condition must be met. Figure 3.2 shows an abstracted overview of the iEpi architecture. Below are the basic components of iEpi.

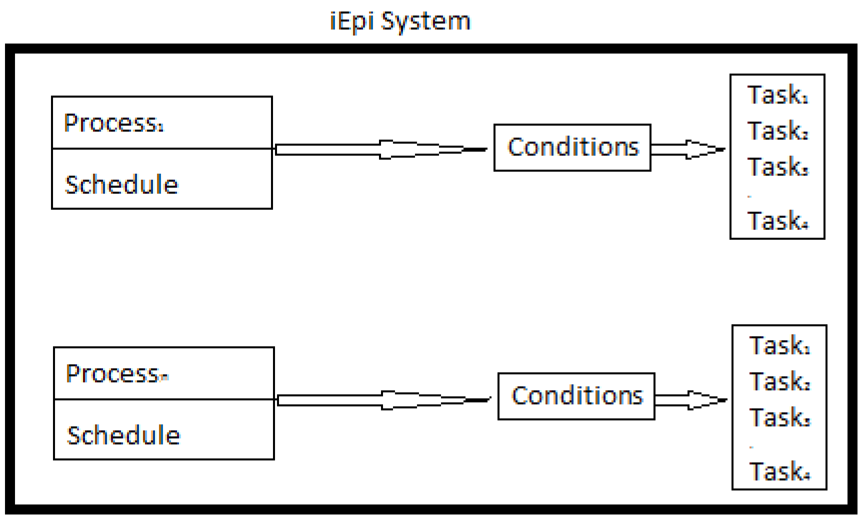


Figure 3.2: An abstracted overview of iEpi architecture.

- Task: The work to be performed. This includes data collection and processing. The tasks are pre-programmed by a developer. Example of tasks includes

```
collect GPS data.
save data.
encrypt data with key.
```

New tasks can be added to the system easily due to the modular design. Task can also be linked together to form a task chain. Tasks can be configured so that the output of one task becomes the input of the other. Below shows an example of a SEDDACCO statement for specifying a task chain:

```
Collect accelerometer data.
convert data to activity.
```

The code above tells iEpi to collect accelerometer data and then pass the data to the convert module to determine the activity of the user.

- Conditions: Blocks of boolean logic, based on a temporary cache of sensor data, which launches a task when true. A task can perform operations on the stored data to produce a condition for another task. Like tasks, conditions are pre-programmed by the programmer and contributed in a modular fashion. Conditions return a boolean value that specifies if the condition was fulfilled. Conditions can be also be linked using AND, OR, and NOT constructs to form more complex predicates. Example:

  ```
  collect gps data if [has moved] and [outside].
  ```

- Schedules: All processes have a schedule which determines the duty-cycle for each process. Example:

  ```
  collect gps data every [2 minutes for 30 seconds].
  ```

3.5 Extended architecture

3.5.1 Background and motivation

When extending the architecture, saving battery life was an important concern. I considered relevant literature and the issues other authors have identified. The most salient issues around battery conservation identified in the literature are outlined below:

- Reduce transition rate [45]: When a phone changes state from idle to active and vice versa, the CPU does more work. A good strategy is to reduce to transition rate by adaptive scheduling. Instead of switching state at a fixed rate, the architecture should be able to specify a delay between the CPU activations. Research has shown that most sensors consume little energy [45]. As an example, I used iEpi [27] to collect accelerometer data continuously for one minute, and store the data in a local SQLite database. No operations other than collection and storage were performed on the data. Figure 3.3 and Figure 3.4 show the results.

  Figure 3.3 shows the CPU usage of the accelerometer sensor when no further processing is performed on the data. We can see that the CPU usage is just above one percent. The spike in the figure is

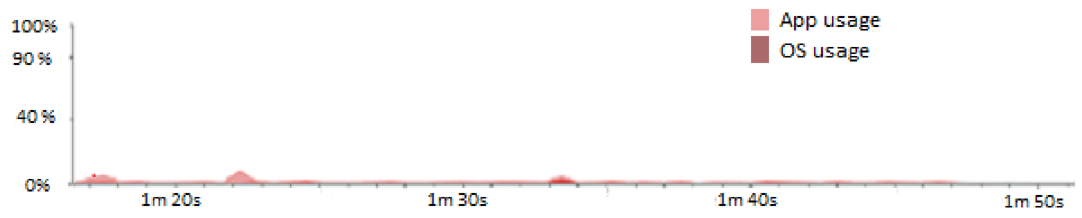


Figure 3.3: CPU usage of accelerometer sensor.

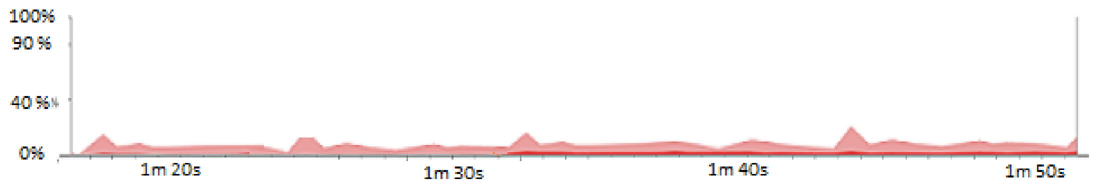


Figure 3.4: CPU usage while saving accelerometer data in a sqlite database.

from the debugger getting a memory snapshot from the mobile device. Looking at Figure 3.4, the CPU usage when data is being stored in a local SQLite data base increases to about 10 percent.

- Use of low-power sensors [4]: Different studies may require different context to trigger a task. To get the user's context, a low-power sensor should be used when possible. Google's API has various functions that use low-power sensor to acquire various contexts like activity recognition, step counting and location.

- Efficient coding [31]: Proper coding practice and test driven development methodology was followed in implementing the architecture. A code coverage tool, EclEmma, was used to find code blocks that used a sensor but fail to unregister the sensor to stop data collection. As seen in Figure 3.5, not stopping data collection from sensors greatly affects the battery life of the device. Another reason for battery consumption is the collection of unnecessary sensor data, for example constantly checking user location when user is asleep [4]. To reduce collection of unnecessary data, the app should be able to recognize user context as it is impossible to reliably predict when a user will be in a certain context like sleeping. iEpi has been configured to allow data post processing on the device and the use of Google API to allow limited context-recognition which was not used in the previous implementation of iEpi.

- Sampling Rate: As can be seen in Figure 3.5, reducing the amount of time the sensors are activated can reduce the energy consumption of the device, but a strict duty-cycle scheme still has the disadvantage of sensors being turned on when the sensor is reporting irrelevant information. As such, there is a need for a flexible measuring regime, as not all sensors are needed at all times.

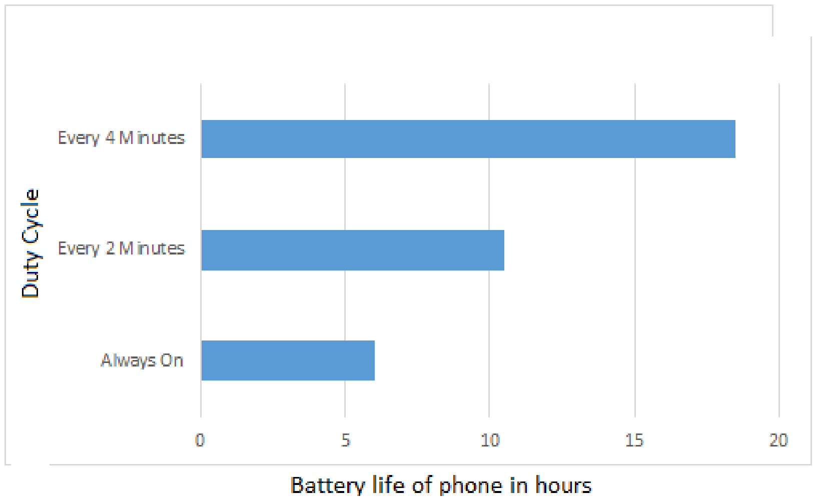


Figure 3.5: Hours until battery is exhausted for different duty cycles.

3.5.2 Adaptive scheduler

Because humans have been shown to follow highly predictable routines [51], we can expect that significant data collection periods would be redundant, and easily representable by previous data. In our extended version, I allow schedules to be altered based on conditions. An experiment in indoor navigation in an institution [43] would require high fidelity data collection from WiFi and location sensors while at the institution, but limited data rates when away, while a study focused on commuting patterns would require detailed location data upon leaving the institution [52]. Clearly, there is a need to configure data collection on a case-by-case, sensor-by-sensor, experiment-by-experiment level. Fortunately, iEpi was designed to be configurable at run-time. By leveraging SEDDACCO, we can extend iEpi's architecture to allow for the specification of conditionally scheduled measurement tasks, and the heuristics underlying them.

In adaptive scheduling, the architecture must allow parallel sensor streams to be articulated based on sensed context, and provide an ability for researchers to specify how the duty cycling should be modified and under what circumstances. This approach allows researchers to operationalize the definition of redundant data through heuristics. If the purpose of the adaptive scheduler is to provide battery savings by eliminating redundant data, then we must allow individual researchers to specify what data is considered redundant, as this will vary on a case by case basis.

I extended the architecture to allow for dynamically reconfigurable independent measurement schedules on each data stream. The job of the adaptive scheduler is to decide the schedule of each process based on certain selected constraints. Figure 3.7 shows the adaptive block.

When specifying a task in SEDDACCO, a schedule must be specified for the task. By default, heuristics are specified in the SEDDACCO language as conditional constructs on sensed data. However, while not shown in Figure 3.7, a more complex scheduler can be added – for example a prediction based scheduler using a Hidden Markov Model – by leveraging the plug and play capabilities of the system through the adaptive block.

Back-off system

Another way that I have tackled the battery issue in iEpi is to introduce a back-off system in iEpi. Back-off systems take advantage of the predictable nature of humans. Humans tend to follow a similar routine: visit similar places on a daily basis like work or being mostly still for most part of the day and moving for shorter periods. The back-off characteristics can be configured using SEDDACCO. The back-off system works by checking if a user-defined condition is met; if the condition is not met, it increases the duty-cycle by a certain amount based on the SEDDACCO configuration. Below is an example of a SEDDACCO program.

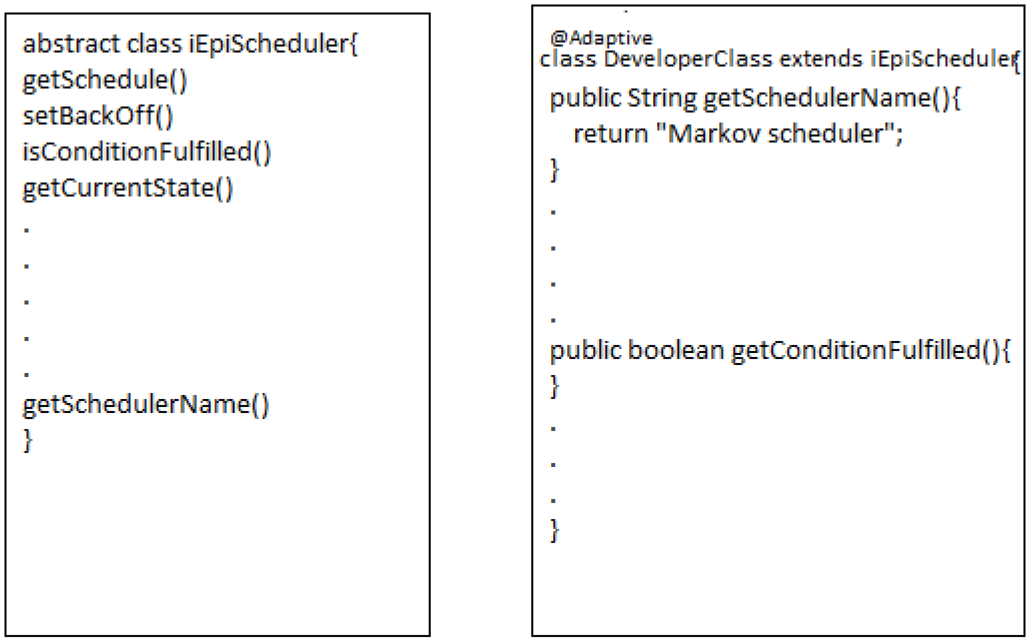
```
collect gps data every [2 minutes for 30 seconds] if
[in university] else back-off[multiple:2 , max:30 mins];
```

The above code tells iEpi to collect GPS data every 2 minutes for 30 seconds if the smartphone detects the user is at the university and otherwise increases the duty-cycle by a multiple of 2, up to a maximum of 30 minutes. If the condition are met, again the duty-cycle returns to 2 minutes.

From the example above, one can see an obvious flaw with the back-off system. It maybe possible for a participant to go to the university and leave again before the next data collection period. However, the back-off system would be applicable in studies interested in a participant's general spatial pattern.

3.5.3 Plug and play architecture

When extending the architecture to enable adaptive scheduling, one major requirement is extensibility - how easy it is to add heuristics to the app. In many cases, patching the entire system on a case by case basis is not practical and, as such, I have implemented a plug and play architecture. From an implementation perspective, each scheduler corresponds to a module. In the plug and play design approach, a module is considered an abstract unit of computation that may have interfaces that define points of interactions to other components in the system like access to sensor data. iEpi defines the boundary of each module and to what part of the system they have access. Modules can also communicate with themselves through a broadcast-receiver architecture. For plug and play capabilities, iEpi exposes a Java abstract class for the adaptive scheduler to which developers can extend. Java reflection capabilities are employed to validate and setup the added scheduler. Ongoing call are made through the methods defined in the abstract class. Before the beginning of each task, iEpi runs each scheduler to determine when each task should start. The scheduler has access to cached sensor data if needed and can also acquire data from other sources like the Internet if the scheduler needs to acquire the current weather conditions.


```
abstract class iEpiScheduler{
getSchedule()
setBackOff()
isConditionFulfilled()
getCurrentState()
.
.
.
.
.
getSchedulerName()
}
```

```
@Adaptive
class DeveloperClass extends iEpiScheduler{
public String getSchedulerName(){
  return "Markov scheduler";
}
.
.
.
.
public boolean getConditionFulfilled(){
}
.
.
.
}
```

Figure 3.6: Implementing an abstract class for the plug and play architecture.

A developer imports the iEpi library JAR file into their project and imports the AdaptiveScheduler abstract class. The main class is annotated using the Java Annotation class. This means developers can have multiple class files. Though only the classes annotated with the *@abstract* keyword will be run by the system. After implementing the required methods, the project should be compiled into a JAR library file and then upload to the server. Once uploaded to the server, a notification is pushed to the device, which then downloads the file. Once downloaded, iEpi creates a dictionary of scheduler names (using the method getSchedulerName already implemented by the developer) and the class, with the scheduler name as the key, and a reference to the class as the value. Scheduler names are unique. On SEDDACCO one can write the following: (*collect temperature data using [schedule name]*). iEpi uses the schedule name to load the required class and calls the getSchedule function which tells the system the schedule of a particular task.

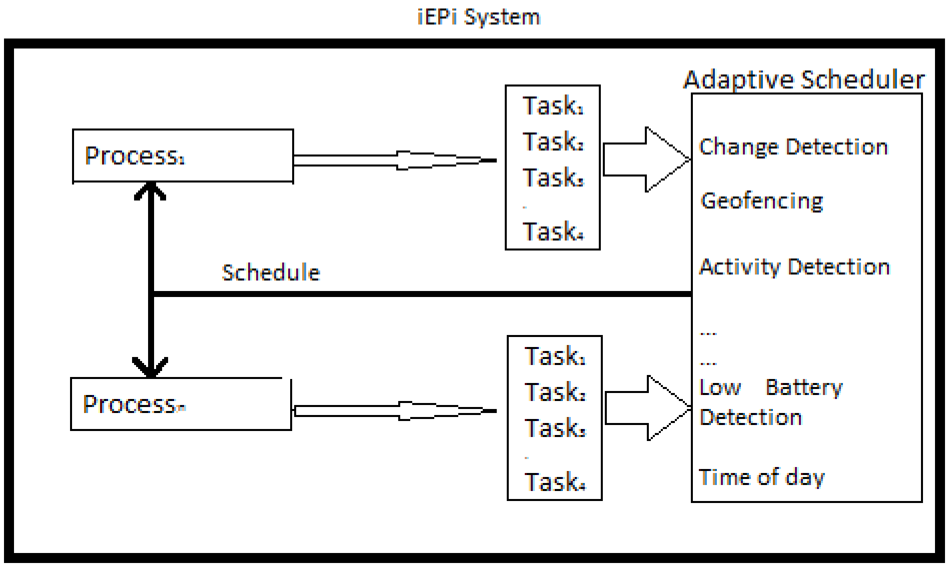


Figure 3.7: An abstracted overview of the extended iEpi architecture.

The new architecture also uses SEDDACCO to specify the system's behaviour to enable rapid establishment and modification of device behavior based on existing functionality. In the extended architecture, one can specify high level constructs of different monitoring techniques. For example:

- ... only if .. else [new duty-cycle]: Select one of two duty-cycles if a condition is true, for example

```
collect gps [every 5 minutes] for 30 seconds only

if [outside] else [every 3 minutes] for 30 seconds;
```

- ... using adaptive scheduler: A more complex adaptive scheduler that can model user activity and schedule, for example

  ```
  collect gps using hmm scheduler;
  ```

One added feature to SEDDACCO is the ability to add parameters to the scheduler, for example:

```
collect Wifi data using Wifi\_only\_scheduler[Wifi\_router:"UofS"];
```

This added feature of adding parameters to the scheduler, creates a more flexible system, as a developer can create a more robust scheduler, for example, a scheduler can be created to only record GPS when around a certain coordinate. The developer will only need to create one scheduler and specify the coordinate using SEDDACCO.

3.6 Use cases

An advantage of this architecture is the capability of measuring only moments of interest, thereby saving battery and at the same time providing quality data. Five use cases are discussed in this section, namely temporal based measurement, spatial based measurement, activity based measurement, event triggered measurement and model triggered measurement.

- Temporal-based measurement: With this architecture one can specify different duty-cycle for different times. This will be of use for people only interested in measuring behaviour at a particular time for example a researcher may be interested in knowing the location of study participants only on work days.
- Spatial-based measurement: One can also specify duty-cycle for different locations, which can be advantageous to researchers interested in measuring participants' behaviour at a particular place, such as measuring students activity pattern when at the university.

- Activity-based measurement: In studies involving physical activity, researchers might be interested in measuring a participant's behaviour only at the times where the participant is active, such as knowing the surrounding temperature whenever a participant is riding a bike. With the extended architecture, this can be accomplished easily with the use of SEDDACCO like so

```
collect temperature data only if [onBike];
```

- Event/Context triggered measurement: In some studies, data might need to be collected only when a certain event occur. An example would be recording data when it is raining, or when a user is playing Pokemon GO.

- Model triggered measurement: The iEpi architecture exposes interfaces which can be used to implement a learning algorithm. This learning algorithm can be used to model a user's behaviour and then trigger measurements based on the model, for example, a researcher can implement a Markov Model of participants spatial pattern and then create a measurement schedule based on the model.

Chapter 4

Experimental Setup

I assessed the efficacy of the system by conducting a pilot study to evaluate the architectural functionality and usability. I also demonstrated that the system can be used in actual real world studies by deploying the system in two Social Science studies. These studies are only described, as the analysis of the data is being conducted by other students and researchers. These studies are presented only as evidence of the practical utility of the work.

4.1 Pilot study

Our experimental setup was meant to provide stylized examples of potential experimental configurations: a primarily indoor space utilization experiment [43], a primarily outdoor study of commuting habits [52], and a study of the spread of contagious disease [18]. The indoor space utilization study is primarily interested in capturing WiFi data for estimating location, as well as any opportunistic GPS readings. Critically, these estimates only need to be high fidelity while participants are at the target location. In the commuting study, the system must capture high fidelity data while the participant is moving, but does not need to record periods of stationary behavior. Finally, in the contagious disease study, changing contact patterns must be captured, as well as the location context for those contacts.

Each of these stylized experimental configurations are based on previous experiments performed using the iEpi system without adaptive sampling, and are readily represented by straightforward heuristics. The space utilization study should reduce measurement frequency when away from the location of interest; this is readily operationalized by examining the SSIDs of visible WiFi routers. The commuting study should decrease measurement frequency when stationary; this is operationalized by examining the logical Android activity stream for the flag 'STILL'. Contact patterns are typically measured with short-distance radio signals like Bluetooth. Changing contact patterns can easily be detected by comparing the list of visible Bluetooth contacts from the previous measurement to the current.

To meet the needs of all three stylized experiments, we measured accelerometer, activity as deduced by the Google activity API, Bluetooth, Battery, GPS and WiFi. To detect movement, we employed the Google activity API, which employs accelerometer, gyroscope and other low power sensors to determine the likely activity state of the participants. Participants were considered to be moving if the Google API returned anything but TILTING and STILL.

I implemented 3 different monitoring schemes for each sensor. The strict duty-cycle was the control condition, as it replicates the existing iEpi scheduling system. The second approach implements a standard back-off strategy: an exponential back-off was triggered whenever a SEDDACCO statement evaluated to true. The final scheduling algorithm implements the same logic as the second, but does not employ back-off, skipping only single measurement period. The three monitoring schemes are detailed below

- **Strict duty-cycle**.
  - GPS: Every 8 minutes for 90 s.
  - Accelerometer, Bluetooth and Wifi: Every 2 minutes for 30 s.
- **Simple Adaptive with Back-off**. Back-off in multiples of 2 until a specified maximum of 16 minutes.
  - GPS: Every 8 minutes if movement and battery $> 10\%$.
  - Wifi: Every 120 s if a designated router is visible within the past 16 minutes and movement.
  - Bluetooth: Every 120 s if there is a change in devices over the last 16 minutes.
  - Accelerometer: Every 120 s for 60 s if movement.
- **Simple Adaptive without Back-off**.
  - GPS: Every 8 minutes if movement and battery $> 10\%$.
  - Wifi: Every 2 minutes if a designated router is visible within the past 16 minutes and movement.
  - Bluetooth: Every 120 s if there is a change in devices in the last 30 minutes.
  - Accelerometer: Every 120 s for 60 s if movement.

I installed our version of iEpi on Samsung Galaxy S4 phones and recruited 30 participants to carry the phone for a period of three weeks, of whom 10 were graduate students, 19 were undergraduates and one was a non-student. Data was collected under the consent of the monitored in accordance with our approval from the university behavioural research ethics review board. The participants were randomly divided into three groups, each of which used a monitoring scheme for one week, with the order condition presentation randomized using a Latin square design. The data collected by the system are encrypted, compressed, and uploaded to a server. There, it is processed by the autoparser, which decrypts the data and inserts it into an appropriate database. Other tools like Tableau, Python, R are used to process, interpret, or otherwise utilize the data.

The goal of this experiment is to

- evaluate the flexibility and utility of the architecture,
- evaluate the degree of battery savings,
- evaluate the quantity and quality of data retrieved.

The flexibility will be evaluated by deploying the system with the different measuring scheme and then reconfiguring the measuring scheme after each week to make sure the system is capable of doing this. In evaluating battery saving, the battery consumption rate for each participant in the different measuring scheme will be calculated. A Friedman's non-parametric test will be used on the data to check if there is a significant difference among the three measuring scheme and which of the scheme saves the most battery. Data quality is difficult to estimate, as it is context dependent. However, the amount of information present in a string can be estimated by examining its information entropy rate. While entropy rate can be difficult to compute for arbitrary strings of data from first principles, it can be readily estimated by performing Lempel-Ziv compression using the LZ78 algorithm [65]. Entropy rate calculations were done using a Python and C++ tool developed internally [46]. Using entropy rate for the streams shows how much information, as defined by the researcher, is present in the stream. For example, if the system is set to record accelerometer data only when a user is moving, there should be fewer data showing the user as been still because the system will only record data when it detects movement.

4.2 Social science studies

Existing literature have shown similar data collection tool with adaptive sampling [3, 8, 41]. However, the current tools have only been used in the context of a single experiment or have not reported moving beyond the prototype stage. In contrast, the architecture presented in this thesis has been deployed in several social science experiments. In this section, I will show the overview of each study and the configuration used in each study. The research questions in this section were proposed by other researchers and thus the final outcome of the studies are not reported in this thesis as the study is yet to be published. However, I am describing the social science studies to demonstrate the breadth of inquiry possible that can be addressed by the system.

4.2.1 SHED6

SHED6 was collected as part of CMPT826 by three groups of graduate students who attempted to answer difference research questions from the same data collection event from the same population. My modified variant of iEpi was used to collect human behavioural data set named the Saskatchewan Human Ethology Dataset 6 (SHED6). iEpi was installed on either the participant's own phone or a study phone. The study lasted for 1 month. Seventy-four participants including undergraduate, graduate students and sessional lecturers were recruited from the University of Saskatchewan. Care was taken to make sure the participants were not colleagues and that no participant had a relationship with any other. They were asked to take an online survey and also provide their schedule together with their time and rough locations for comparison purposes. They were also made aware of privacy concerns and the purpose of data collection. The study was approved by the University Behavioural Ethics Review Board. Participants were asked to carry the phones with them at all times during the day and to charge them at night. Several research interest were simultaneously studied using the tool. The tool allowed simultaneous several experiments to be run simultaneously on the same device. Table 4.1 shows the study summary.

Demographics	74 participants
Duration	30 days
Purpose	• Dietary habit of University students and detecting eating gestures using smart-phone sensor data. • Activity times in different Neighbourhood Active Living Potential areas. • Improving indoor localization using atmospheric pressure.
Contextually data collection rules	• Location constraint: if movement is detected or location accuracy is below 20 metres, collect data every 4 minutes for 1 minutes otherwise backof with multiple of 2. • Accelerometer constraint: if movement is detected and at the University of Saskatchewan, collect data continuously otherwise backof in multiple of 2. • Pressure constraint: if location accuracy is more than 20 metres, collect data every 4 minutes for 30 seconds. • Temperature constraint: if location accuracy is more than 20 metres, collect data every 4 minutes for 30 seconds. • Battery: always collect 2 battery records every 4 minutes. • Survey constraint: if time is between 6pm and 8pm or 7am and 9am and the survey is yet to be answered.
Outcomes	Confidential until publication.
Realized architecture benefits	• Battery consumption. The architecture's scheduler allowed participants to use their personal phone, unlike previous studies which were restricted to secondary phones because of battery consumption. • The architecture's scheduling flexibility allowed multiple studies with different configurations to be run simultaneously. • The plug and play feature allowed multiple changes to be made while the study was in progress.

Table 4.1: Saskatchewan Human Ethology Dataset 6 (SHED6) summary.

The research questions that was studied using the SHED6 data are

- Dietary Habits of University of Saskatchewan Students,
- Activity Times in Different Neighbourhood Active Living Potential (NALP) Areas,
- Improving Indoor Localization Using Atmospheric Pressure.

Dietary habits of University of Saskatchewan Students

Healthy eating is an important factor in maintaining a healthy lifestyle. Bad eating habits have been linked with obesity. Other literature suggests that poor nutrition is also associated with adverse mental health [53]. Healthy eating could help control the prevalence of food related abnormalities such as obesity and mental health issues. Data could be leveraged to provide healthy nutritional intervention programs to educate the population and especially the young adults about cultivating dietary habits with regard to healthy food sources.

This study focused on understanding and analyzing dietary habits of the University of Saskatchewan students with a view of providing nutritional interventions to encourage healthy dietary attitudes among young people.

iEpi gathered sensor data from participants and uploaded the collected data to a database server over a trusted Internet connection. At the end of each day between the hours of 7:00 PM and 10:30 PM, a survey popped up, asking the participants what they ate for the various meals of the day and when they ate them. Participants were allowed to skip entering survey data for any meal of the day that was missed.

Activity times in different NALP areas

Physical inactivity and obesity are major public health problems. One promising strategy in fighting obesity and increasing physical activity is the creation of healthy environments [63]. Studies have shown that physical inactivity, along with subsequent related health outcomes, is associated with neighbourhood characteristics [5, 6]. Understanding characteristics of neighbourhoods that facilitate active living is an emerging challenge for researchers in increasing the prevalence of healthy lifestyles among the population. Studying how a neighbourhood can influence the health of individuals and populations involves identifying environmental variables associated with physical inactivity. This is critical in developing effective interventions targeting environmental and policy change aiming to address the societal burden of sedentary lifestyles. Identifying those neighbourhood characteristics involves the development of a reliable and valid method that capture properties of the built environment. Researchers [10, 14, 15] have used NALP to measure the likelihood of active living in a neighbourhood. NALP looks at the characteristics in a neighbourhood that can be associated with physical activity, providing a limited cross-sectional snap shot of a population.

iEpi was used as a tool for sensor data collection. Sensor data related to individual participants including location and the physical activity (such as walking, running, biking) undertaken by the participant in that location. Using ArcGIS, NALP scores were assign to the location and activity of each of individual participants during the duration of the study. Statistical analysis was used to determine if daily active living is affected by the neighbourhood in which a participant resides.

Improving indoor localization using atmospheric pressure

Mobile devices are widely used for location services such as GPS. GPS is the most popular technology to identify exact location outdoors but within large buildings its functionality becomes ineffective. Although indoor localization systems have been developed, to improving their accuracy and responsiveness remains a challenge. Some smartphones are equipped with atmospheric sensors that allow people monitor environmental temperature, humidity and air pressure. This study presents a method that helps to improve indoor localization by utilizing the atmospheric pressure sensor.

iEpi was used to collect WiFi, GPS and atmospheric pressure data. The WiFi data collected was filtered and was fed into the University of Saskatchewan GIS System, which was further processed in order to return the location of the closest routers. After getting the location of the routers, a particle filter was used to determine the location of a participant around campus. The atmospheric pressure was used to calculate the approximate altitude of the participant.

4.2.2 Food security

Neighbourhood conditions have been studied and found to be a contributing factors in promoting health disparities [29, 48]. It is widely accepted that low income neighborhoods are disproportionately affected by increased rates of morbidity, mortality and adverse health outcomes [11, 12]. Several studies have reported the tendency of poor and minority neighborhoods to have fewer supermarkets which offer a larger variety of affordable and healthy foods compared to smaller convenience stores [35]. This is of importance due to the emergence of "food deserts" in many low-income neighborhoods that result from the absence of a supermarket. This study aimed to reveal how households in low income neighbourhoods feed themselves. The research examined the household food practices of families - the food thay they eat and how they acquire food in the context of alternative food networks and why residents do or do not access the various community-based food interventions being offered in the community.

In this study, iEpi was used to collect sensor-based traces of human behaviour, including location, contact patterns, activity levels and survey data. The photo diary functionality within iEpi was used to collect additional information about their food by asking participants to take a photograph of everything they eat over a three-day period once a month. Participants were also asked to photograph all food purchases that they make during one randomly selected week each month. Table 4.2 shows the summary of the study.

Demographics	17 families
Duration	3 months
Purpose	Reveal how households in disadvantaged neighbourhoods feed their families in the context of alternative food network and obesogenic food environments.
Data collected	• Location constraint: always collect location data every 8 minutes for 1 minute. • Battery: always collect 2 battery records every 4 minutes. • Survey constraint: it varies with participant and day of the week. • Photo: user triggered.
Outcomes	Confidential until publication.
Realized architecture benefits	• Battery consumption. • Ease of use. Health researcher was able to configure iEpi using Seddacco. • The plug and play feature of the architecture allowed multiple changes to be made while the study was in progress. • Scheduling was used to send reminders at appropriate times and contexts.

Table 4.2: Summary of data collected for food security study

Chapter 5

Results

This chapter discusses the results from the pilot experiment which demonstrates the technical advantages of the system. The results show the data collected from various sensors with the 3 monitoring schemes described in the previous chapter. This chapter also shows a summary of data collected from an actual social science study to demonstrate the functionality of the system beyond a proof of concept.

5.1 Results from pilot experiment

To determine the impact of the measurement protocols, I analyzed the collected data in four ways. First, I examined relative compliance levels amongst participants to determine overall data quality. Second, I evaluated the relative battery life of each condition. Third, I examined the number of records reported for each scheduler, and the resulting structure by post-processing data returned in the strict condition with the two adaptive algorithms. Finally, I estimated the information quality by computing the information entropy rate for the data. In subsequent comparisons in this section, the strict algorithm is referred to as Non-adaptive (N), the adaptive with back-off algorithm as Adaptive 1 (A1) and the adaptive without back-off as Adaptive 2 (A2). When boxplots are used, the whiskers represent the outer (0-25 and 75-100) quartiles, and the box represents the inner (25-50 and 50-75) quartiles.

5.1.1 Compliance rate

Participant adherence to the study protocol is important for understanding the degree to which the results might be biased by poor data. Participants were asked to take a study phone wherever they went for the period of the study and to make sure to keep it charged. To estimate the compliance rate of participants, I use the charging pattern of their smartphones, labeling them as discharging and charging from either the main power or via a computer USB. Figure 5.1 shows the charging pattern of the participants. A perfect participant will have approximately two thirds of the total number of possible records in the discharging state and another third in the charging state corresponding to the instructions to carry the phone at all times and charge it at night. For example, participant 20 in figure 5.1 shows the phone has a higher number of records in the charging state compared to discharging which indicates the participant may have left the phone plugged in for a prolonged period of time.

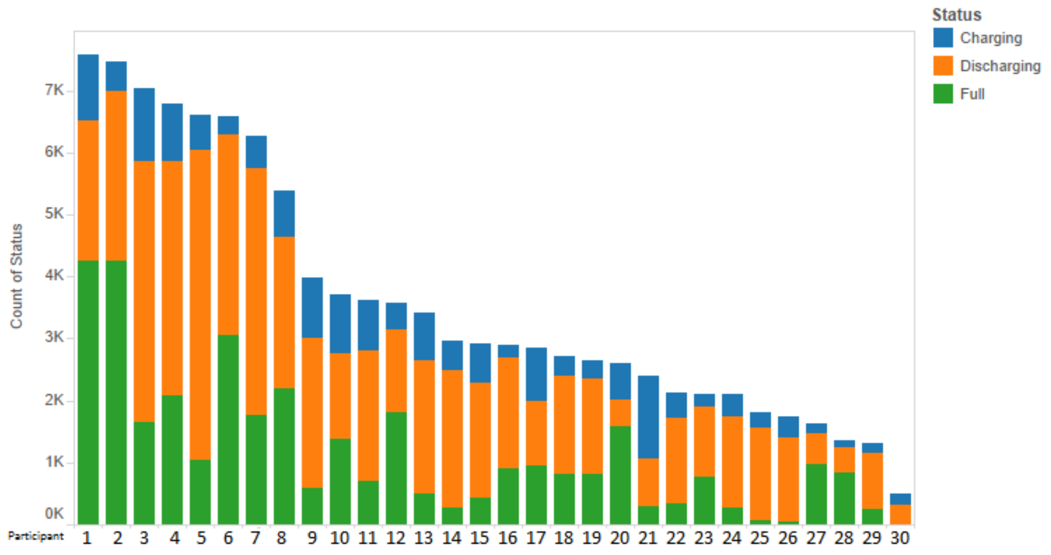


Figure 5.1: Compliance Rate.

iEpi records battery data every 4 minutes for all measuring schemes. The total possible records is 7560 for each participant. As is evident from the figure, there is strong heterogeneity between participant compliance levels, with some reporting close to 100% of possible data and others reporting less than 20% of possible data. Possible reasons for the heterogeneity between compliance levels are due to forgetfulness or neglect, with the phone not always being charged at night, and software or hardware malfunction. Participant's compliance rate tend to reduce with time. Figure 5.2 shows the amount of battery data gathered per week over the 3 weeks period. I can see the number of data begin to reduce after the first week.

5.1.2 Power consumption

The primary motivation for adaptive sampling is to reduce the battery consumption. Reducing redundant data both directly saves power by leaving the phone in sleep mode, and indirectly by reducing the volume

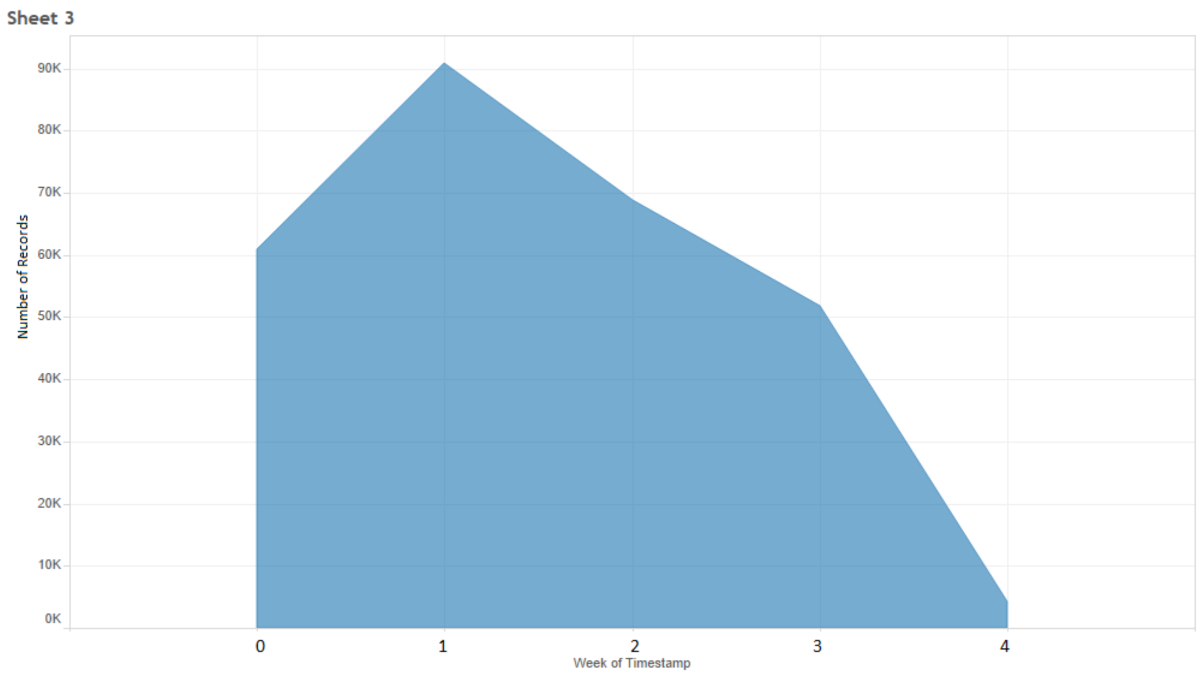


Figure 5.2: Number of battery records with time.

of data that requires wireless backhaul. I measured the average battery consumption for each regime by dividing the percentage of battery lost in each discharging state by the total time spent in the discharging state as shown in Figure 5.3. I expect that the within subjects design will account for differences in app use. The non-adaptive case has a greater median, total and span of battery use across participants, as expected. While the non-back-off adaptive scheme has a higher median consumption rate, the span of battery consumption between A1 and A2 is similar. As expected, the back-off scheme has a higher probability of a lower consumption rate than the other two, as it can enter states in which measurements happen relatively rarely.

A Friedman test on the battery consumption data from the 3 duty-cycle scheme resulted in a p-value less than 0.0001, indicating that there is an overall statistically significant difference among the three measurement schemes.

5.1.3 WiFi

In the adaptive cases, WiFi was limited when not at the target location. Figure 5.4 shows the total records in each regime. As expected, the total number of WiFi records was numerically higher in the non-adaptive

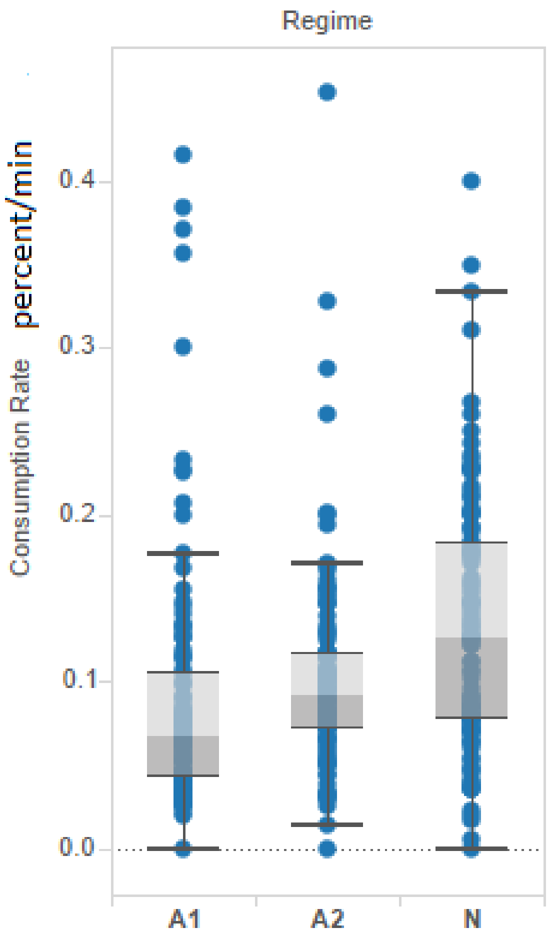


Figure 5.3: Battery consumption rate for each monitoring regime

case.

Figure 5.5 shows a pie chart of the ratio of times a university router was in proximity to when no university router was in proximity. As expected from the monitoring schemes, A2 has the greatest ratio of university router records, because A2 is set to only record WiFi data when the phone is around the University of Saskatchewan. A1 on the other hand, has a higher number of non-university routers than A2 because A1 is set to always record data even when not near the university; however, it will collect data less frequently due to the back-off algorithm resulting in a higher ratio of non-university router than A2. The strict measuring regime, N, has no constraints on when and where to collect data resulting in a larger ratio of non-university routers to university routers.

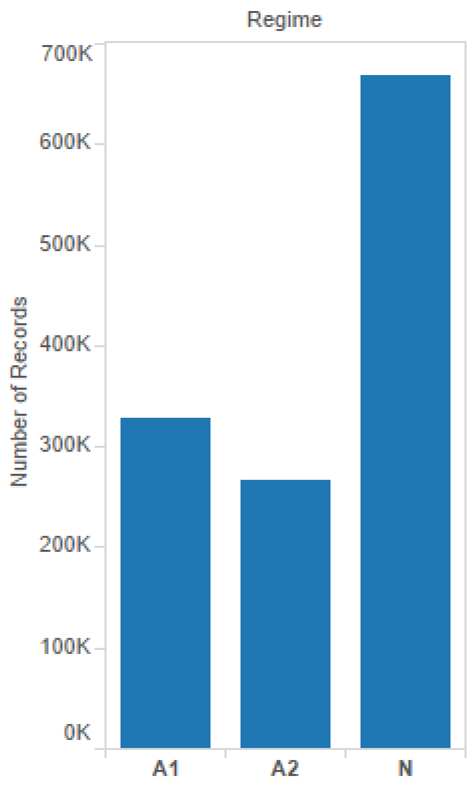


Figure 5.4: Count of WiFi records in each regime.

5.1.4 Location

Location records are often over-measured, as almost all people remain stationary for at least a third of the day while sleeping or eating, and most remain in the same locations (work, home or school) for prolonged periods of time [49]. Figure 5.6, shows the total number of location records for each condition, where A1 and A2 have less than one tenth the number of records of N. A single trace on a map (the streets where removed from the map to protect the participant's privacy) from one participant is shown in Figure 5.7, where the adaptive algorithms were applied to the data collected in the non-adaptive case. This particular trajectory starts near the top of the image. Because the participant has remained in that location for a significant amount of time, A1 is fully backed off, and does not start measuring the trajectory until later than A2. If movement patterns are important for the research question, A1 seems like a poor choice; however, if the research is primarily interested in destinations instead of routes, then A1 remains viable.

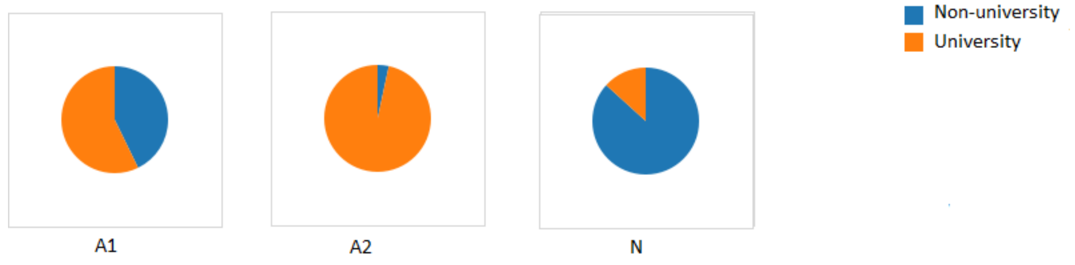


Figure 5.5: Pie charts showing the ratio of number of times, for each measure scheme, a university router was in proximity to when no university router was in proximity.

5.1.5 Entropy rate

The entropy rate can be used to estimate the information density of a stream of data. Increasing entropy rate corresponds to increasing information density, and is desirable for selective downsampling. While this metric cannot replace expert domain knowledge for particular experiments and streams, it can be used to determine if the information density of adaptive regimes is greater than or comparable to the non-adaptive cases. I approximated the entropy rate of the location, battery and activity streams, as shown in Figure 5.8. Running Friedman's test to compare the three measurement schemes resulted in a p-value of less than 0.0001, indicating that there is an overall statistically significant different among the three measurement schemes.

The battery entropy rate, as in Figure 5.8, corresponds to the behavior of the participants, as battery measurements were not adapted. As expected, the entropy rate for all three are generally similar, although in case A2, the median is drawn down by outliers, who likely left their phones plugged in, leading to repeated charging measurements. As expected from Figure 5.7, A2 has the greatest mobility entropy rate, as it eliminates long repetitions of the same location (which have low entropy rates), but captures mobility patterns (which have higher entropy rates). A1 and N have lower entropy rates, but A1 is still on average higher than N. Finally, I compared the entropy rates of strings made from activity labels, and again, as anticipated by Figure 5.7, A2 has the highest entropy rate, for similar reasons to the location entropy rate. N has a low entropy rate given the dominance of repeated "STILL" labels in the string. A1 has a lower rate because it misses many of the short transitions due to overly long back-off periods, while A2 eliminates repetition, but captures transitions.

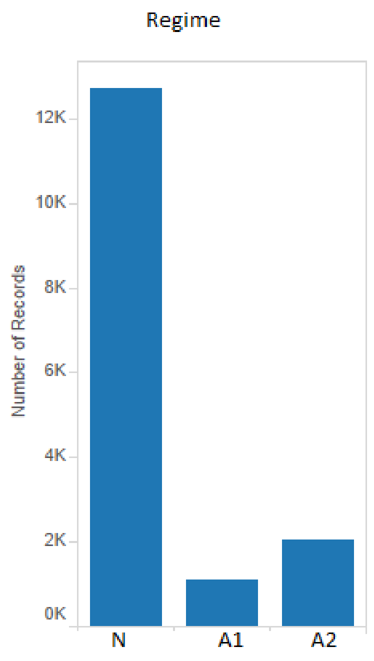


Figure 5.6: Count of locations records in each regime.

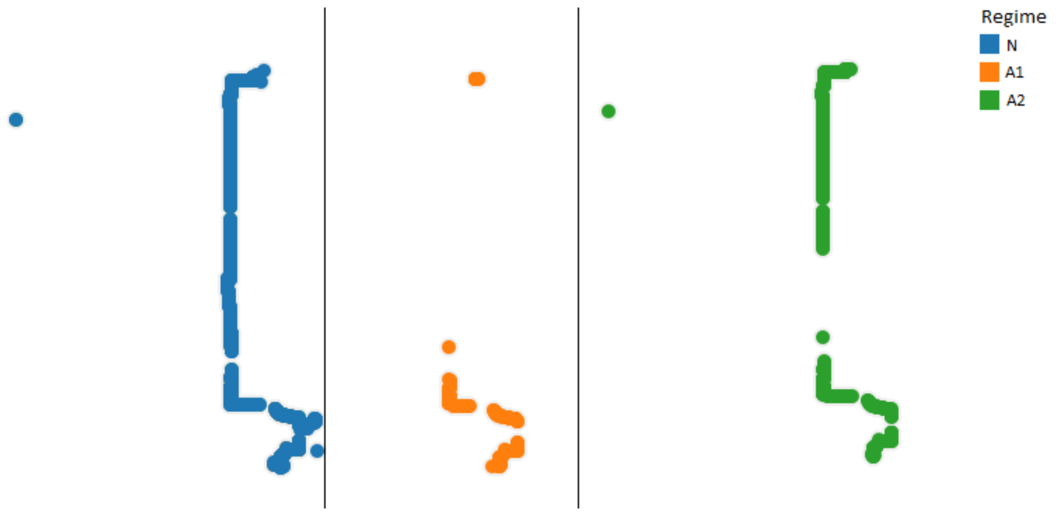


Figure 5.7: A participant's trajectory in each measuring scheme for a single day.

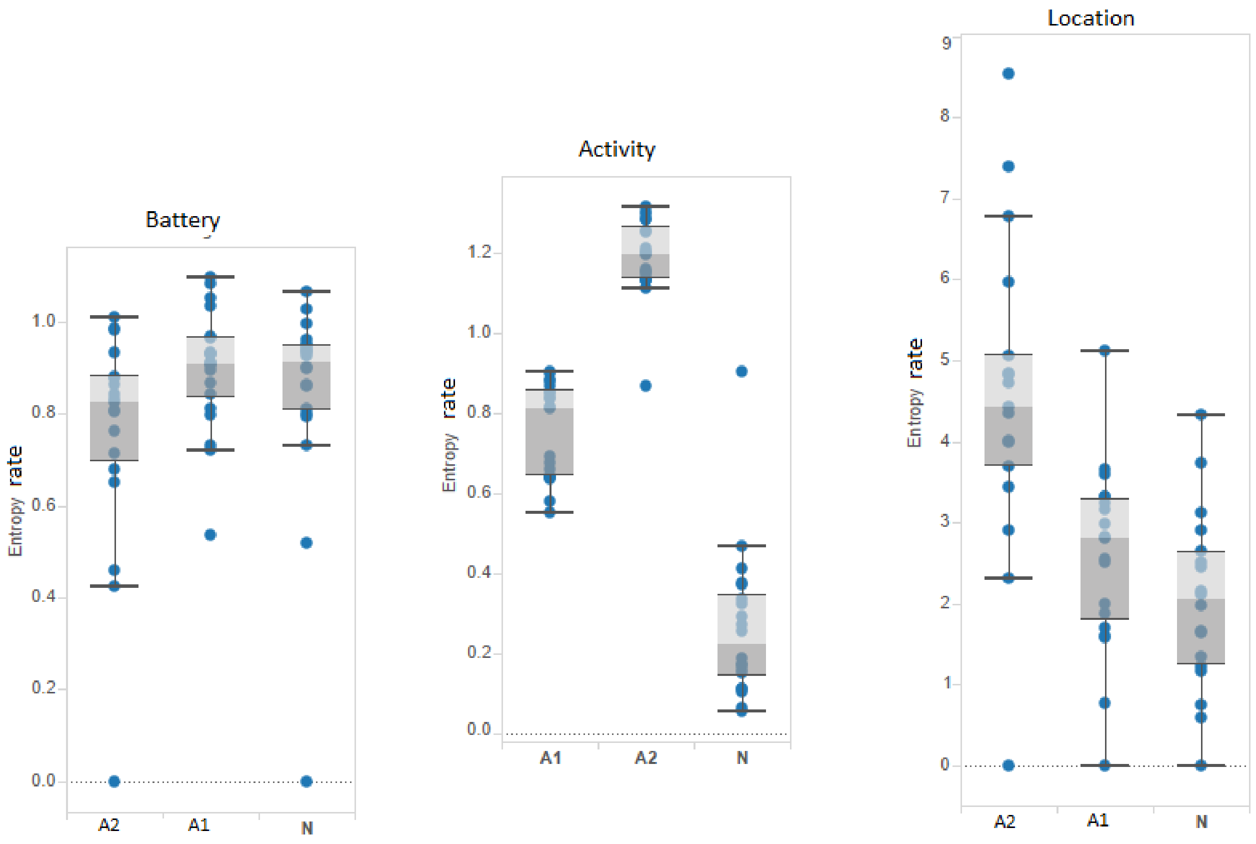


Figure 5.8: Entropy rate of all participants' battery, activity and location data.

5.2 Data summary from deployment

In addition to this pilot data, iEpi has successfully been used for 2 data collection studies. The first study involves the food acquisition and consumption habit of 17 families. These families were monitored for a period of 6 months. The research goal was to reveal how households in disadvantaged neighbourhoods feed their families in the context of a series of important community-based food interventions unfolding in obesogenic food environments. iEpi was used to record their mobility and activity patterns. The analysis aimed to identify the relationship between mobility pattern and food acquisition/consumption. The second study, known as SHED6, involves the collection of data from 30 students for a period 30 days. There were multiple research questions been studied using the data collected. While the data is still being investigated and studied by the researchers, I have provided here a brief summary of the data collected as shown in Table 5.1 and Table 5.2 to show that system was successfully deployed in actual health and social science studies. This section do not show the results or the outcome of the experiments.

Data summary from food security study		
Date	Number of Records	Average records per participant
Accelerometer	44,565,153	1,937,615
GPS	14,67,785	66,717
WiFi	1,205,792	54,808
Picture diary	686	35

Table 5.1: Number of records for each data acquisition task.

Data summary from SHED6		
Date	Number of Records	Average records per participant
Accelerometer	65,996,176	1,534,794
GPS	52,493	1,312
WiFi	4,72,191	11,805
Bluetooth	219,813	5,111
Pressure	12,674,841	294,763
Temperature	1,270,085	29,536
Survey records	1844	686

Table 5.2: Number of records for each data collection task.

5.3 Chapter Summary

In this chapter, I showed the result from the pilot study and the data summary from two social science studies. The results from the pilot study aimed to show the architectural functionality and usability. Energy savings was demonstrated through analysis of energy consumption rates, which were lower across both adaptive conditions. I also showed the result from the adaptive regime for Wifi and GPS which had less than 50 percent of the total number of data gotten by the strict duty cycle regime. With such reduced quantity, I was able to demonstrate that important information was captured in the adaptive cases by employing a recent metric – the information entropy rate of the sensor streams – I was able to show an increase in information density for the non-back-off adaptive algorithm, and better or equal information density for the back-off algorithm, allowing us to conclude that redundant data was removed. Based on the post-processed analysis of the non-adaptive data using the adaptive algorithms, and the increase in information entropy rate for the adaptive algorithms, I believe that I have demonstrated that the data quality is sufficient. Lastly, I showed that the system was successfully used in two human behaviour study by providing the summary of the data collected in both studies.

Chapter 6

Conclusion

I have described a novel system and architecture for specifying adaptive patterns based on logic embedded in a SEDDACCO configuration file. This system was evaluated in a three week user study that established ease of use, energy efficiency, and data quality of the implemented adaptive approaches, and employed in two large studies demonstrating the system's utility.

6.1 Summary

The thesis proposes a new, flexible, and energy-efficient architecture for human behavior measurement using smartphones. A flexible system for defining adaptive measurement heuristics was described as an extension to an existing system. The system was evaluated in a user study to determine the impact on data quality and battery life of different heuristics applied separately to different sensor streams. Using both traditional methods and the more novel information entropy rate metric, we were able to demonstrate that the adaptive schemes saved significant battery consumption. This thesis also showed that the system was successfully used in two human behaviour studies. The first study involved monitoring the food consumption habits of 17 families and the second study, known as SHED6, involved the collection of data from 74 students for a period of 30 days. The versatility of the system was tested as iEpi was configured to record specific sensor data at specific instances. This work represents an important step in extending the capabilities of smartphone measurement platforms, which will impact both future system designs and researchers using similar systems for scientific inquiry.

6.2 Conclusions

In this work, I did not directly evaluate the usability of the architecture, which would require additional experiments using techniques from the Human Computer Interaction and Software Engineering literatures. However, I was able to successfully deploy a set of adaptive measurement heuristics on several sensor streams, providing some confidence that our first goal – ease of use – is plausible. Our second goal – achieved energy savings – was clearly demonstrated through analysis of energy consumption rates, which were lower across both adaptive conditions. Data quality is more difficult to quantify, but I was able to demonstrate that important information was captured in the adaptive cases. Employing a recent metric – the information entropy rate of the sensor streams – I was able to show an increase in information density for the non-back-off adaptive algorithm, and better or equal information density for the back-off algorithm, allowing us to conclude that redundant data was removed. Based on the post-processed analysis of the non-adaptive data using the adaptive algorithms, and the increase in information entropy rate for the adaptive algorithms, I believe that I have demonstrated that the data quality is sufficient.

Comparing the back-off and non-back-off cases provides additional insights. There were incremental energy savings in the back-off case, but at the expense of transient data. For researchers primarily interested in broader behaviors, for example, where participants shop, simple adaptive sampling with back-off could be a viable choice. However, for researchers interested in transitions, for example studying how transit schedules change participants' shopping habits, the back-off case would miss critical transient behaviors, compared to simple adaptive sampling without back-off.

The following significant contributions were made:

- **Flexible Architecture**: While previous work has examined the impact of adaptive sampling, this thesis is the first to my knowledge to provide an architecture where adaptation can be configured at run time through scripting.
- **Usage Profiling**: The results provide valuable insight into the trade-offs in employing a classic back-off-style scheduler, which may miss important transition periods.
- **Entropy rate as a metric**: To the best of my knowledge, this thesis is the first to employ the

information entropy of the downsampled stream as a performance metric for adaptive sampling of human behaviour. This important measure could benefit other researchers attempting to evaluate the efficacy their algorithms.

- **Plug and Play architecture**: The system provides a modular architecture where more functionality can be added. This helps designers easily design multiple studies and also allows for change during a study. The architecture also allows communication between module using the Android broadcast capability.
- **SEDDACCO**: SEDDACCO has been extended to allow specification of multiple scheduling schemes.

The capability of the system has successfully been used for two studies. The first study involves the food acquisition and consumption habit of 17 families. These families were monitored for a period of 3 months. iEpi recorded their mobility and activity patterns over the 3 months period. The photo diary functionality in iEpi was used to store pictures and audio from the participants. The second study, known as SHED6 involves the collection of data from 30 students for a period 30 days. The versatility of iEpi was tested as iEpi was configured to record specific sensor data at specific instances, like pressure data when at the university. Participants were allowed to use their own phone, something that was not possible with earlier version of iEpi due to battery constraints. During the 30 day study, participants reported their phone lasting close to a full day.

With these contributions, I believe this work could benefit engineers and computer scientists trying to design similar systems, geographers, civil engineers and social scientists trying to better understand their data, and data analysts trying to better quantize the impact of sampling.

6.3 Future Work

iEpi has been shown to be more than a research curiousity by the number of studies that have used it [27]. However, there is still work to be done on to optimize and extend its operation, including memory and CPU optimization, as well as usability, security and functional improvements.

- Cache optimization: iEpi temporarily stores data so different conditioned statements can be processed.

For example, a scheduler might want to check if the participant has moved in the last 10 minutes, requiring a cache check. These caches uses SQLite to store the data. However, SQLite on Android has usability and reliability issues. A more efficient is possible [20, 21]. This new solution has the potential to reduce the overhead associated with the existing cache system, and to make context recognition and condition evaluation possible at the moment changes occur.

- Improve concurrency: iEpi relies on multiple processes running simultaneously. There have been instances where multiple processes try to use the same resources, thereby causing missing data and crashes. This also affects the energy usage of the device: a process waiting for a resources to be released keeps the device from going into an idle state.

- Energy profiling: The study done in this paper did not take into account the energy used by other apps. A full energy usage profile would better inform developers on parts that need optimization.

- Improve database storage: iEpi has a central task that stores all data. The task periodically flushes the data in the cache database into a file. Currently, the data are saved one at a time rather than in batch. Single insertion in Android is slow and inefficient. A better approach will be for iEpi to accumulate the data in memory and store directly into a file instead of first storing it into a database.

- Snooze mode: iEpi allows users to stop iEpi from recording their data for defined time. However, the approach used for this functionality is constantly using the cpu to check if iEpi is in a snooze state, thereby draining the device battery.

- Parallel tasks: Parallelization is increasingly important in computing applications. The current architecture requires a custom scheduler to achieve task parallelization i.e., allowing data output from one task to multiple sources. A better approach would be to expose this functionality in Seddacco, for example the following syntax could allow for more sophisticated parallel operation.

```
<task chain>
then simultaneously
<task chains>
(done; | when done <task chain>)
```

For example:

```
collect accelerometer data
every 5 minutes for 30 seconds
then simultaneously
update map icon;
stream location to server;
done.
```

- Events: Events are useful when attempting to detect context changes. The current architecture requires an adaptive scheduler to detect and respond to events, which is less flexible. A better approach would be to enable specification in SEDDACCO. For example

```
when <condition expression> becomes <boolean>
<parallel task chains>
done.


when <outside> and <raining> becomes true
issue ‘‘bad weather survey’’;
sample accelerometer for 30 seconds;
done.
```

In addition to the architectural change, in the future, deploying this system in a larger study could evaluate the impact over a longer term, with a larger participant pool. Evaluation by users of the suitability of the extensions to SEDDACCO will be necessary. Finally, combining the adaptive sampling regime with learning algorithms, would allow the parameters of the adaptive sampling to adapt to different users or situations.

References

[1] Aanensen, D. M., Huntley, D. M., Feil, E. J., and Spratt, B. G. (2009). Epicollect: linking smartphones to web applications for epidemiology, ecology and community data collection. *PloS One*, 4(9):e6968.

[2] Abbate, S., Avvenuti, M., Bonatesta, F., Cola, G., Corsini, P., and Vecchio, A. (2012). A smartphone-based fall detection system. *Pervasive and Mobile Computing*, 8(6):883–899.

[3] Becker, R. A., Caceres, R., Hanson, K., Loh, J. M., Urbanek, S., Varshavsky, A., and Volinsky, C. (2011). Route classification using cellular handoff patterns. In *Proceedings of the 13th International Conference on Ubiquitous Computing*, pages 123–132.

[4] Ben Abdesslem, F., Phillips, A., and Henderson, T. (2009). Less is more: energy-efficient mobile sensing with senseless. In *Proceedings of the 1st ACM workshop on Networking, Systems, and Applications for Mobile Handhelds*, pages 61–62.

[5] Booth, S. L., Sallis, J. F., Ritenbaugh, C., Hill, J. O., Birch, L. L., Frank, L. D., Glanz, K., Himmelgreen, D. A., Mudd, M., and Popkin, B. M. (2001). Environmental and societal factors affect food choice and physical activity: rationale, influences, and leverage points. *Nutrition Reviews*, 59(3).

[6] Brownson, R. C., Baker, E. A., Housemann, R. A., Brennan, L. K., and Bacak, S. J. (2001). Environmental and policy determinants of physical activity in the United States. *American Journal of Public Health*, 91(12):1995–2003.

[7] Carroll, A. and Heiser, G. (2010). An analysis of power consumption in a smartphone. In *Proceedings of the 2010 USENIX annual technical conference*, volume 14, pages 21–21.

[8] Chon, Y., Kim, Y., Shin, H., and Cha, H. (2014a). Adaptive duty cycling for place-centric mobility monitoring using zero-cost information in smartphone. *IEEE Transactions on Mobile Computing*, 13(8):1694–1706.

[9] Chon, Y., Talipov, E., Shin, H., and Cha, H. (2014b). SmartDC: Mobility prediction-based adaptive duty cycling for everyday location monitoring. *IEEE Transactions on Mobile Computing*, 13(3):512–525.

[10] Corburn, J., Osleeb, J., and Porter, M. (2006). Urban asthma and the neighbourhood environment in New York City. *Health & Place*, 12(2):167–179.

[11] Cubbin, C., Hadden, W. C., and Winkleby, M. A. (2000). Neighborhood context and cardiovascular disease risk factors: the contribution of material deprivation. *Ethnicity & Disease*, 11(4):687–700.

[12] Deaton, A. and Lubotsky, D. (2003). Mortality, inequality and race in American cities and states. *Social Science & Medicine*, 56(6):1139–1153.

[13] Eagle, N. and (Sandy) Pentland, A. (2006). Reality mining: sensing complex social systems. *Personal and Ubiquitous Computing*, 10(4):255–268.

[14] Gauvin, L., Riva, M., Barnett, T., Richard, L., Craig, C. L., Spivock, M., Laforest, S., Laberge, S., Fournel, M.-C., and Gagnon, H. (2008). Association between neighborhood active living potential and walking. *American Journal of Epidemiology*, 167(8):944–953.

[15] Gray, L. and Leyland, A. H. (2009). A multilevel analysis of diet and socio-economic status in Scotland: investigating the 'Glasgow effect'. *Public Health Nutrition*, 12(09):1351–1358.

[16] Gummadi, R., Gnawali, O., and Govindan, R. (2005). Macro-programming wireless sensor networks using Kairos. In *Proceedings of the 2005 Distributed Computing in Sensor Systems*, pages 126–140.

[17] Hashemian, M., Knowles, D., Calver, J., Qian, W., Bullock, M. C., Bell, S., Mandryk, R. L., Osgood, N., and Stanley, K. G. (2012). iEpi: an end to end solution for collecting, conditioning and utilizing epidemiologically relevant data. In *Proceedings of the 2nd ACM International Workshop on Pervasive Wireless Healthcare*, pages 3–8. ACM.

[18] Hashemian, M. S., Stanley, K. G., and Osgood, N. (2010). Flunet: Automated tracking of contacts during flu season. In *Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt), 2010 Proceedings of the 8th International Symposium on Modeling and Optimization in Mobile Ad Hoc Networks*, pages 348–353.

[19] Hnat, T. W., Sookoor, T. I., Hooimeijer, P., Weimer, W., and Whitehouse, K. (2008). Macrolab: a vector-based macroprogramming framework for cyber-physical systems. In *Proceedings of the 6th ACM conference on Embedded network Sensor Systems*, pages 225–238.

[20] Kang, S., Lee, J., Jang, H., Lee, H., Lee, Y., Park, S., Park, T., and Song, J. (2008). Seemon: scalable and energy-efficient context monitoring framework for sensor-rich mobile environments. In *Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services*, pages 267–280.

[21] Kang, S., Lee, J., Jang, H., Lee, Y., Park, S., and Song, J. (2010). A scalable and energy-efficient context monitoring framework for mobile personal sensor networks. *IEEE Transactions on Mobile Computing*, 9(5):686–702.

[22] Kim, D. H., Kim, Y., Estrin, D., and Srivastava, M. B. (2010). Sensloc: sensing everyday places and paths using less energy. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, pages 43–56.

[23] Kirovski, D., Oliver, N., Sinclair, M., and Tan, D. (2007). Health-os:: a position paper. In *Proceedings of the 1st ACM SIGMOBILE International Workshop on Systems and Networking Support for Healthcare and Assisted Living Environments*, pages 76–78.

[24] Kjærgaard, M. B., Bhattacharya, S., Blunck, H., and Nurmi, P. (2011). Energy-efficient trajectory tracking for mobile devices. In *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services*, pages 307–320.

[25] Knowles, D. (2015). The design and use of a smartphone data collection tool and accompanying configuration language. Master's thesis.

[26] Knowles, D., Stanley, K. G., and Osgood, N. D. (2014a). Seddacco: An extensible language in support of mass collection of health behavior data. In *Proceedings of the 2014 ACM SIGKDD Workshop on Health Informatics.*

[27] Knowles, D. L., Stanley, K. G., and Osgood, N. D. (2014b). A field-validated architecture for the collection of health-relevant behavioural data. In *Proceedings of the 2014 IEEE International Conference on Healthcare Informatics*, pages 79–88.

[28] Kothari, N., Gummadi, R., Millstein, T., and Govindan, R. (2007). Reliable and efficient programming abstractions for wireless sensor networks. In *Proceedings of the 2007 ACM SIGPLAN Notices*, volume 42, pages 200–210.

[29] Lee, C. (2005). Environmental justice. *Environmental Health: From Global to Local.*, pages 170–196.

[30] Lin, K., Kansal, A., Lymberopoulos, D., and Zhao, F. (2010). Energy-accuracy trade-off for continuous mobile device location. In *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, pages 285–298.

[31] Liu, Y., Xu, C., and Cheung, S.-C. (2013). Where has my battery gone? finding sensor related energy black holes in smartphone applications. In *Proceedings of the 2013 IEEE International Conference on Pervasive Computing and Communications*, pages 2–10.

[32] Madden, S., Franklin, M., and Hellerstein, J. A tiny aggregation service for ad hoc sensor network. In *Proceedings of the 5th Symposium on Operating Systems Design and Implementation*, pages 131–146.

[33] Miller, J. S., Dinda, P. A., and Dick, R. P. (2009). Evaluating a basic approach to sensor network node programming. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, pages 155–168.

[34] Min, C., Yoo, C., Lee, Y., and Song, J. (2011). Healthopia: Towards your well-being in everyday life. In *Proceedings of the 4th International Symposium on Applied Sciences in Biomedical and Communication Technologies*, pages 108–113.

[35] Morland, K., Wing, S., Roux, A. D., and Poole, C. (2002). Neighborhood characteristics associated with the location of food stores and food service places. *American Journal of Preventive Medicine*, 22(1):23–29.

[36] Mukhopadhyay, S. C. (2015). Wearable sensors for human activity monitoring: A review. *IEEE Sensors Journal*, 15(3):1321–1330.

[37] Naik, B. A. and Chavan, R. (2015). Optimization in power usage of smartphones. *International Journal of Computer Applications*, 119(18):7–13.

[38] Naqvib, N. Z., Kumar, A., Chauhan, A., and Sahni, K. (2012). Step counting using smartphone-based accelerometer. *International Journal on Computer Science and Engineering*, 4(5):675–681.

[39] Osgood, N. D., Paul, T., Stanley, K. G., and Qian, W. (2016). A theoretical basis for entropy-scaling effects in human mobility patterns. *PloS One*, 11(8):e0161630.

[40] Paek, J., Kim, J., and Govindan, R. (2010). Energy-efficient rate-adaptive GPS-based positioning for smartphones. In *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, pages 299–314.

[41] Paek, J., Kim, K.-H., Singh, J. P., and Govindan, R. (2011). Energy-efficient positioning for smartphones using cell-id sequence matching. In *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services*, pages 293–306.

[42] Peltonen, E., Lagerspetz, E., Nurmi, P., and Tarkoma, S. (2016). Where has my battery gone?: A novel crowdsourced solution for characterizing energy consumption. *IEEE Pervasive Computing*, 15(1):6–9.

[43] Petrenko, A., Bell, S., Stanley, K., Qian, W., Sizo, A., and Knowles, D. (2013). Human spatial behavior, sensor informatics, and disaggregate data. In *Proceedings of the 2013 International Conference on Spatial Information Theory*, pages 224–242.

[44] Podsakoff, P. M., MacKenzie, S. B., Lee, J.-Y., and Podsakoff, N. P. (2003). Common method biases in behavioral research: a critical review of the literature and recommended remedies. *Journal of Applied Psychology*, 88(5):879–903.

[45] Priyantha, B., Lymberopoulos, D., and Liu, J. (2011). Littlerock: Enabling energy-efficient continuous sensing on mobile phones. *IEEE Pervasive Computing*, 10(2):12–15.

[46] Qian, W., Stanley, K. G., and Osgood, N. D. (2013). The impact of spatial resolution and representation on human mobility predictability. In *Proceedings of the 2013 International Symposium on Web and Wireless Geographical Information Systems*, pages 25–40.

[47] Sandhu, R. S. and Samarati, P. (1994). Access control: principle and practice. *IEEE Communications Magazine*, 32(9):40–48.

[48] Sexton, K. (2000). Socioeconomic and racial disparities in environmental health: is risk assessment part of the problem or part of the solution? *Human and Ecological Risk Assessment*, 6(4):561–574.

[49] Shannon, C. E. (1949). Communication theory of secrecy systems. *Bell Labs Technical Journal*, 28(4):656–715.

[50] Smith, G., Wieser, R., Goulding, J., and Barrack, D. (2014). A refined limit on the predictability of human mobility. In *Proceedings of the 2014 IEEE International Conference on Pervasive Computing and Communications*, pages 88–94.

[51] Song, C., Qu, Z., Blumm, N., and Barabási, A.-L. (2010). Limits of predictability in human mobility. *Science*, 327(5968):1018–1021.

[52] Stanley, K., Bell, S., Kreuger, L. K., Bhowmik, P., Shojaati, N., Elliott, A., and Osgood, N. D. (2016). Opportunistic natural experiments using digital telemetry: a transit disruption case study. *International Journal of Geographical Information Science*, 30(9):1853–1872.

[53] Tanskanen, A., Hibbeln, J. R., Tuomilehto, J., Uutela, A., Haukkala, A., Viinamäki, H., Lehtonen, J., and Vartiainen, E. (2001). Fish consumption and depressive symptoms in the general population in finland. *Psychiatric Services*, 52(4):529–531.

[54] Thiagarajan, A., Ravindranath, L., LaCurts, K., Madden, S., Balakrishnan, H., Toledo, S., and Eriksson, J. (2009). Vtrack: accurate, energy-aware road traffic delay estimation using mobile phones. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, pages 85–98.

[55] Tsapeli, F. and Musolesi, M. (2015). Investigating causality in human behavior from smartphone sensor data: a quasi-experimental approach. *EPJ Data Science*, 4(1):1–24.

[56] Tu, Y.-H., Li, Y.-C., Chien, T.-C., and Chou, P. H. (2011). Ecocast: interactive, object-oriented macroprogramming for networks of ultra-compact wireless sensor nodes. In *Proceedings of the 2011 10th International Conference on Information Processing in Sensor Networks*, pages 366–377.

[57] Van der Spek, S., Van Schaick, J., De Bois, P., and De Haan, R. (2009). Sensing human activity: GPS tracking. *Sensors*, 9(4):3033–3055.

[58] Vazquez-Prokopec, G. M., Stoddard, S. T., Paz-Soldan, V., Morrison, A. C., Elder, J. P., Kochel, T. J., Scott, T. W., and Kitron, U. (2009). Usefulness of commercially available GPS data-loggers for tracking human movement and exposure to dengue virus. *International Journal of Health Geographics*, 8(1):1–8.

[59] Wark, T., Crossman, C., Hu, W., Guo, Y., Valencia, P., Sikka, P., Corke, P., Lee, C., Henshall, J., and Prayaga, K. (2007). The design and evaluation of a mobile sensor/actuator network for autonomous animal control. In *Proceedings of the 6th International Conference on Information Processing in Sensor Networks*, pages 206–215.

[60] Werner-Allen, G., Lorincz, K., Ruiz, M., Marcillo, O., Johnson, J., Lees, J., and Welsh, M. (2006). Deploying a wireless sensor network on an active volcano. *IEEE Internet Computing*, 10(2):18–25.

[61] White, E., Basford, L., Birch, S., Black, A., Culham, A., McGoff, H. J., Lundqvist, K. O., Oppenheimer, P., Tanner, J., and Wells, M. (2015). Creating and implementing a biodiversity recording app for teaching and research in environmental studies. *The Journal of Educational Innovation, Partnership and Change*, 1(1):1–26.

[62] Whitehouse, K., Zhao, F., and Liu, J. (2006). Semantic streams: A framework for composable semantic interpretation of sensor data. In *Proceedings of the 2006 European Workshop on Wireless Sensor Networks*, pages 5–20.

[63] World Health Organization (2016). Obesity and overweight. fact sheet N 311.

[64] Yun, X., Bachmann, E. R., Moore, H., and Calusdian, J. (2007). Self-contained position tracking of human movement using small inertial/magnetic sensor modules. In *Proceedings of the 2007 IEEE International Conference on Robotics and Automation*, pages 2526–2533.

[65] Ziv, J. and Lempel, A. (1978). Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*, 24(5):530–536.