International Workshop on Sensing for App Phones

PhoneSense 2010

November 2nd, 2010

Zurich, Switzerland

International Workshop on Sensing for App Phones (PhoneSense 2010)

Nov. 2 , 2010
Zurich, Switzerland

App phones are enabling the delivery of personalized sensing applications across a wide variety of users and to very large geo distributions. A number of recent developments give momentum to this new field of research, including: the proliferation of embedded sensors in open programmable smartphones; the ease at which researchers and developers can distribute new applications using vendor specific "app store" delivery channels (e.g., Apple AppStore, Android Market, Microsoft Mobile Marketpace, Nokia Ovi); and, finally, the emergence of the mobile computing cloud. The combination of three drivers is creating new opportunities in mobile phone sensing not seen before.

Emerging applications on apps phones and cloud can sense, mine, and learn human behaviors and intentions to provide personalized feedback and persuasion. Example application domains include mobile advertisement, social networking, healthcare, entertainment, education, safety and business.

This workshop promotes exchanges among academic and industrial researchers in related areas, such as sensing, mobile computing, data management, data mining, machine learning, inference, incentive modeling, persuasion feedback, user experience with app delivery channels for large-scale deployment, and privacy. The foci are on position papers, novel ideas, and in-progress work on system architecture, enabling technologies, and emerging applications.

The topics of interest include, but are not limited to:

- Applications
- Sensing and machine learning techniques
- Experience with app store delivery systems and large scale deployment
- Mobile cloud and sensing Interface and interaction between phones and humans
- Mining big sensor data
- Persuasion models and techniques to close the loop with users
- Privacy and sensor data
- Participatory and opportunistic sensing paradigms
- Activity recognition and subjective sensing
- Programming models
- Experiment and campaign design

Workshop Organizers

Co-Chairs

Andrew Campbell (Dartmouth College)
Jie Liu (Microsoft Research)

Technical Program Committee

Tarek F. Abdelzaher (UIUC)
Frank Bentley (Motorola)
Gaetano Borriello (University of Washington)
Romit Roy Choudhury (Duke)
Tanzeem Choudhury (Dartmouth College)
Sunny Consolvo (Intel Research)
Deepak Ganesan (UMass Amherst)
Bhaskar Krishnamachari (USC)
Dimitrios Lymberopoulos (Microsoft Research)
Mani B. Srivastava (UCLA)

Workshop Program

8:55 Opening Remarks

9:00 - 10:00 Keynote: Transitioning from Locations to Semantics and Services
Andreas Savvides (Yale University)

10:00 - 10:30 Coffee break

10:30 - 12:00 Session 1: Platforms and Services

Session Chair: Andrew Campbell

ContextDroid: an Expression-Based Context Framework for Android
Bart van Wissen, Nicholas Palmer, Roelof Kemp, Thilo Kielmann, Henri Bal (Vrije Universiteit, The Netherlands)

An Integrated Monitoring System for Smart Phones
Christopher Miller, Christian Poellabauer (University of Notre Dame, USA)

BSMX - A Prototype Implementation for Distributed Aggregation of Sensor Data
Sascha Schnaufer, Stephan Kopf, Wolfgang Effelsberg (University of Mannheim, Germany)

EERS: Energy Efficient Responsive Sleeping on Mobile Phones
Nissanka Priyantha, Dimitrios Lymberopoulos, Jie Liu (Microsoft Research, USA)

12:00 - 13:00 Lunch

13:00 - 14:30 Session 2: Contexts and Applications

Session Chair: Behrooz Shirazi

Pocket, Bag, Hand, etc. - Automatically Detecting Phone Context through Discovery
Emiliano Miluzzo (Dartmouth College, USA); Michela Papandrea (University of Applied Sciences of Southern Switzerland Switzerland, Switzerland); Nicholas D Lane (Dartmouth College, USA); Hong Lu (Dartmouth College, USA); Andrew Campbell (Dartmouth College, USA)

A Comfort Measuring System for Public Transportation Systems Using Participatory Phone Sensing
Cheng-Yu Lin (Academia Sinica, Taiwan), Ling-Jyh Chen (Academia Sinica, Taiwan), Ying-Yu Chen (Academia Sinica, Taiwan), Wang-Chien Lee (The Pennsylvania State University at University Park, USA)

A Smartphone Based Fall Detector with Online Location Support
Gokhan Yavuz, Mustafa Kocak, Gokberk Ergun, Hande Ozgur Alemdar, Hulya Yalcin, Ozlem Durmaz Incel, Cem Ersoy (Bogazici University, Turkey)

Smartphone Tracking for Historical Event Data Retrieval
Vikram P. Munishwar, Nael Abu-Ghazaleh (State University of New York at Binghamton, USA)

14:30 - 15:00 Coffee break

15:00 - 16:10 Sesson 3: Privacy

Session Chair: Jun Yang

NRC Data Collection and the Privacy by Design Principles
Imad Aad (Nokia Research Center, Switzerland), Valtteri Niemi (Nokia, Switzerland)

Catching Cheats with Interactive Proofs: Privacy-preserving Crowd-sourced Data Collection Without Compromising Integrity
Akshay Dua, Nirupama Bulusu, Wu-chang Feng (Portland State University, USA)

Opportunistic privacy preserving environmental noise pollution monitoring
Luca Becchetti, Luca Filipponi, Andrea Vitaletti (DIS Sapienza Universita' di Roma, Italy)

16:10 - 17:30 Panel: "Business Aspects of Phone-Sensing"

Research has been developing numerous examples of sensing, mining, and learning human behaviors. Will these examples find their way into business practice? What are the corresponding business models? Which are the technical challenges yet to be solved?

Host: Florian Michahelles

Panel
Juha Laurila, Laboratory Director at Nokia Research Center Lausanne
Matthias Sala, CEO of Gbanga
Roman Bleichenbacher, Codecheck

17:30 - 17:35 Closing Comments

ContextDroid: an Expression-Based Context Framework for Android

Bart van Wissen, Nicholas Palmer, Roelof Kemp, Thilo Kielmann and Henri Bal
Vrije Universiteit
De Boelelaan 1081A
Amsterdam, The Netherlands
{bvwissen, palmer, rkemp, kielmann, bal}@cs.vu.nl

Application	Purpose	Context
Always Prompt	notify when to leave	calendar, location
SongDNA	song recognition	sound
Hoccer	information exchange	location, movement
Screebl	power saving	movement, orientation
PhotoShoot [3]	duel game	movement, orientation
iNap	notify when close to destination	location

Table 1: Context Aware Applications Seen in ADC2

ABSTRACT
In this paper we describe our work on ContextDroid, a framework for Context Aware applications on Android powered smartphones. This framework is designed to provide application developers with the services required to easily build Context Aware applications with an eye towards reducing energy consumption of context monitoring, especially as multiple context aware applications are run on the same device.

1. INTRODUCTION
We are currently living in the Information Age, an era characterized by an abundance of information and the presence of many technological devices interacting with this information. One of these devices, the *smartphone*, is of particular interest for personal context information.

Where the first generation phones, were only equipped with radios for communication, modern smartphones have many sensors. They can, for instance, sense location, nearby devices, movement of the phone using an accelerometer and/or gyroscope, proximity to the users skin, and even compass orientation. These devices are often used to store personal information such as contacts and calendar information, but they can also be used to retrieve information from web services such as weather, traffic, and news services as well as run advanced applications.

Due to the opening of centralized markets, we have seen an explosion in the number of third party applications for smartphones. A similar rise on smaller scale occurred for applications that take advantage of context information available on the phone in order to make the phone behave in a smarter way. In Table 1 we list applications that use context information which participated in the recent Android Developers Challenge for innovative smartphone applications.

Although we see an increase in the number of context aware applications, writing such applications can be a complex task, especially when the information comes from many different types of sensors, each with their own unique programming interface.

Furthermore, when multiple context aware applications are used in combination, it is likely that more than one will monitor the same sensors. This can result in highly inefficient use of the device's resources as duplicate analysis of sensor data can not be combined. If programmers would use an easy to program centralized framework for this, they would not only save development time and be able to develop more interesting applications, but in addition the phone's resources could be used more efficiently, resulting in longer battery life for the user.

In this paper we discuss which properties are required for such a framework, and present our ongoing work on the design and implementation of *ContextDroid* a Context Framework running on the Android platform.

The contributions of this paper are:

- We show that there is a need for a Context Framework to ease programming of context aware applications and reduce inefficient use of resources
- We present the requirements for a Context Framework for smartphones
- We present the design and implementation of ContextDroid, a Context Framework that fulfills the necessary requirements.
- We demonstrate this framework with a simple babymonitor application

The remainder of this paper is organized as follows. In Section 2, we describe the requirements for a Context Framework derived from our analysis of the problem. In Section 3 we describe the design of ContextDroid in a bottom-up fashion, starting from the sensor and ending at context expressions. Section 4 goes into detail about important aspects or our implementation. We then demonstrate ContextDroid with an application that can be used to monitor a sleeping baby in Section 5 and conclude by giving an overview of related work in Section 6, future work in Section 7 and conclusions in Section 8.

2. REQUIREMENTS
In our research project, called Interdroid, we strive towards creating an environment for smartphones in which truly smart (possibly distributed) applications can easily be constructed. We believe that smart usage of context information is of key importance for next generation smart applications. Ideally such a smart application will automatically adapt its behavior towards the actual context of the smartphone and its user. For instance, during an important meeting incoming phone calls should be directly sent to voicemail without ringing, but if a user's partner calls three times in a row, there might be something even more

important and then the phone should notify the user. As another example, when close to the supermarket during opening hours, the shopping list application should inform you that you have to buy milk. However, it should not do so if you have a meeting scheduled for which you, according to your location, have to hurry to be on time.

Today, when application programmers write such an application, they have to use the available programming interfaces of the platform to access several context sensors. Accessing the position information from the GPS is totally different from looking into the calendar, or getting the current sound level. For each and every situation the programmer has to write a new – possibly complex – piece of code that will get the desired context information. They also have to write the logic which will allow them to combine these pieces of information into smart behaviors.

We believe that a generic framework for the gathering and evaluation of context information will greatly reduce development time for such applications, but only if it meets the following requirements:

- *Usable*: The system has to provide a framework that enables application developers to very easily make their applications context-aware, even if context-awareness is not the main point of their application. Access to different context elements should be uniform.
- *Efficient*: Running on portable devices with limited battery-life means that attention to energy consumption has to be paid from the beginning.
- *Extensible*: Writing intelligent context-knowledge gathering routines is a complex task that is best left to programmers that have specific knowledge of the sensors. Thus the system should enable third party programmers to easily add components to the system.
- *Portable*: Software built using the framework has to be usable on a large number of devices.

3. DESIGN

In this section we describe the different elements of our context framework, *ContextDroid*, which meets the aforementioned requirements. We explain the design of our context framework beginning with our model of context information, and continuing with how we gather this information, and then how expressions can be formed around them and finally to how those expressions will be evaluated. We also detail how ContextDroid deals with energy usage and Quality of Service through service level requests and how these mechanisms can be extended.

3.1 Context Entities

The devices context, as known to ContextDroid, consists of *context entities*. A context entity is a collection of information. For example, the user's current location, expressed in latitude, longitude, and altitude, is a context entity. The current state of that entity is determined by context sensors. Context sensors are programs that gather information from a particular source (hardware sensors, internal memory, files, the internet) and deliver it as a context entity to ContextDroid.

We split context entities into two categories:

- *synchronous entities*, the state of which can be read in real time. Example: the current time.
- *asynchronous entities*, the state of which is 'pushed' into the system. The service then saves this state in a history so that applications can obtain the state of such an entity at a certain point in time.

The state of an entity may be unknown at any time, at which point its value will be `null`.

3.2 Context Entity Readings

We call the state of a context entity during a certain period in time a *context entity reading.* A reading consists of a value that represents the state of the entity, a timestamp, and a time of expiration. The reading is valid from its timestamp up to its time of expiration. A new reading effectively invalidates and replaces the previous reading from the timestamp of the new reading, even though the previous reading's expiration time may indicate that it is still valid.

This approach was chosen because the system may not always be able to provide new readings in time. Readings must thus have an expiry time, which can be chosen by the sensor providing the reading. Highly dynamic context entities will in general have a shorter time of validity than context entities that hardly ever change.

3.3 Context Conditions

Typically an application designer does not want to use the raw context entity readings, but rather wants to evaluate them according to a function with application specific parameters. For instance the raw location is of much less interest to a developer than the evaluation of whether the distance to the supermarket is less than 50 meters. Therefore, in order to make programming easy, our framework provides *context conditions.*

A context condition is defined by a boolean evaluator function, a number of constant parameters for the evaluator function and a description of what context information should be used as input.

Evaluator functions are functions which take one variable and a number of constants as arguments and return a boolean value. To formalize this, we designed an object called EvaluatorInputSelector. This object contains settings that together define which value should be passed to the evaluator function. Those settings are:

- a Context Entity identifier
- a "value path" describing which part of the entity to take in case it is a complex value (see below)
- a time span
- a selection mode

The context entity identifier uniquely identifies the context entity.

3.3.1 Value Path

Some context entities have values that are actually a set of key/value-pairs, or a list of values, or even a list of sets of key/value pairs. We may want an evaluator function to use a part of that complex data structure as input to the evaluation, for instance we may only be interested in the altitude from the GPS sensor, so we need a way to describe which part of a given entity an expression is needed. For this, we use a path-string, which allows regular expression like selections of parts of values. For brevity we don't go into details of the selection mechanism.

3.3.2 Time Span

The time span specifies a window in the history of the state of the context entity. The effect of this window depends on the mode, but in general it determines how much history to take into account when evaluating the condition.

3.3.3 Selection Mode

The selection mode can be one of the following:

- ANY: The condition is true if the evaluator returns true for any of the values inside the window.
- MEAN: The condition is true if the evaluator returns true for the mean value calculated over the values inside the window. This can only be used for numeric values.
- MAXIMUM: The condition is true if the evaluator returns true for the highest value inside the window. This can only be used for values that implement the Comparable interface.
- MINIMUM: Like the former, but with the lowest value instead of the highest.
- ALL: The condition is true if the evaluator returns true for all values currently inside the window.

3.4 Evaluating a Context Condition

Evaluating a context condition consists of the following steps. First a part is selected out of a context entity's history. Then, from every reading inside this history, one or more atomic values are selected. Depending on the mode, in case of ANY, all values are passed to a *evaluator* in reverse order (latest first) until one value is found for which it returns true, or in case of MINIMUM, MAXIMUM or MEAN, first a value is calculated and then it is passed to the evaluator.

The settings of the EvaluatorInputSelector can have a big impact on performance. When no history is used, only one value has to be evaluated, and it only has to be done whenever a new value comes in, making the condition very cheap to evaluate. On the other hand, when the time span is very long, the number of readings inside it is high, and a mode like MINIMUM or MEAN is used, evaluating the condition can become a time- and energy consuming operation.

We apply several evaluation strategies to optimize the evaluation. When mode ANY is used, if the evaluator returns true for a value that just entered the window, we can be sure that it will be true for as long as the value is inside the window. We can calculate at what moment it will move out of the window, and make sure that we do not evaluate before that moment.

When mode MINIMUM is used, we only have to re-evaluate when the latest minimum value moves out of the window, or when a new value enters the window that is lower than the last one.

A similar optimization can be done for MAXIMUM. The most problematic mode is MEAN, because it changes continuously, even when no new values come in. Since we cannot evaluate it continuously, we chose to automatically reevaluate it when no new value has come in for a certain period.

3.5 Evaluators

The *evaluator* is a simple interface with one method named evaluate, which takes a variable number of parameters. ContextDroid includes a number of predefined evaluators. A few of these are:

- ==, >=, >, <, <=: These evaluators act as their names imply. They all take two arguments and compare them.
- regexp: This evaluator takes a string and a pattern as arguments and returns true if the string matches the pattern.
- distance within: This evaluator takes two pairs of coordinates and a radius as arguments, and returns true if the distance between the two pairs of coordinates is less than the specified radius. This can be used to create "proximity alert" style conditions.

3.6 Expressions

In order to provide a way for applications to react to certain conditions in the context, as mentioned in the requirements, applications should be able to describe that set of conditions to the service in a formal way. For this, our framework uses *context expressions*. A context expression is a boolean expression in which the axioms are the context conditions on context entities.

Since most conditions are a combination of several subconditions, we need a way to combine conditions into expressions. We have chosen to use a tree structure because every logical formula can be expressed as a tree. In such a tree, a conjunction is an AND node with two children, a disjunction is an OR node with two children, and a NOT node has only one child whose result is inverted. The leaf nodes are the atoms, which are specific context conditions.

In addition to being expressive and intuitive to use the tree structure also adds the possibility of short-circuit evaluation. In an expression A OR B, evaluation of B can be skipped as long as A is true. Future work with our framework will include optimizing such short circuiting based on energy consumption to minimize the energy consumption of evaluating the total framework, for instance by turning off high energy consumption sensors such as the gps when in conjunction with a low energy sensor such as time.

3.7 Quality of Service: Service Level Requests

Different applications may have different demands in terms of information that has to be available. One application may be dependent on an entity which is highly dynamic, such as the microphone level, and may want to react to changes quickly. Another application may only want to check periodically if some WiFi network is available.

Because battery life is limited, mobile operating systems are designed to put the phone into a sleep mode whenever possible. To perform any readings for our context service however, the phone has to be awake. Our background service should thus ideally not perform any readings when they are not really necessary.

Thus, we decided to have the applications tell the service the minimum service level that they require to operate correctly. They do this by issuing a service level request. In principle, an application can request any service level, but no guarantees are given by the service. The service will deliver a best effort that matches the requested service level as closely as possible. A service level consists of a list of context entities, each with a number of parameters. These parameters include:

- whether or not the entity's sensor has to be active
- the minimum frequency at which readings of that entity should be performed
- the amount of history that should be kept
- entity-specific requirements

Upon receipt of a service level request, the service adjusts its settings according to the request. When multiple applications issue service level requests, for each setting the "highest" setting is chosen. All those highest settings together form the composite service level. Consequently, whenever

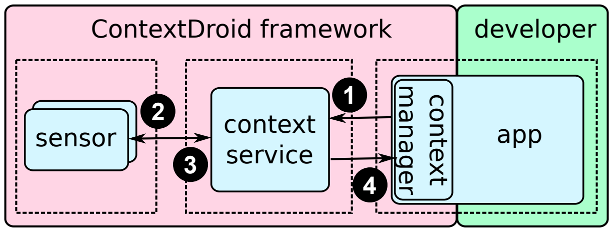


Figure 1: Overall ContextDroid Structure. Each dashed box is a separate process. Service requests are passed from the application using the manager to the service (1) and forwarded to the subsequent sensor (2) via IPC. Sensors push readings back to the service (3), which evaluates them and notifies the application if needed via a broadcast (4).

a request is cancelled, the service level is restored to the highest level without that request.

3.8 Extensibility

It is very important that the default set of available context information made available by our system is easy to extend with new entities. To create a new context entity, basically two conditions must be fulfilled:

- The entity itself must be declared, giving it an identifier and a data type.
- A sensor must be implemented to provide readings for that entity.

Note that a single sensor can (and in most cases will) actually provide readings for multiple related entities. Implementing a new entity may thus also mean adding its implementation to an existing sensor, provided the developer has access to its source code.

4. IMPLEMENTATION

Now that we have seen the design of the expression based context framework ContextDroid, we turn our attention to the implementation. We selected the open source Android mobile platform as target platform for our implementation. Android has several application components that match well with the requirements and design of ContextDroid. Android allows for long running background processes (*services*) which fit particularly well for both the Sensors and the ContextManager, a feature unavailable on the popular iPhone platform.

Figure 1 shows the different components of ContextDroid's implementation. A client-service architecture was chosen to enable efficient sharing of knowledge between multiple applications that run simultaneously.

Android's AIDL interfaces have been used to enable transportation of objects in so-called *parcels* between client and server. The application is linked with *ContextManager*, a helper class that facilitates communication between the application and the ContextDroid service. The ContextManager sets up a connection to the service interface, sends service level requests and installs listeners to receive broadcasts from the server.

The service launches and maintains connections to the Sensors, which are in themselves Android services with their own interfaces. Sensors connect back to the ContextDroid service to 'push' context knowledge, which effectively makes a two-way binding between a sensor and the ContextDroid service.

The ContextDroid service provides the applications with context knowledge by means of *Broadcast Intents*. Broadcast intents are Android's way of broadcasting information to applications system-wide. New readings as well as expressions of state-transitions are broadcast this way.

4.1 The ContextService

The ContextService is the heart of the Context Framework. Its main responsibilities include:

- maintain a shared knowledge base of context information
- process updates to the context information provided by Context Sensors
- provide an interface and act as a mediator between applications, the context knowledge base and the service level manager

The context service maintains a history of readings for a specific context entity. The amount of history is set by the Service Level and can be changed dynamically.

The service level manager is also included in the context service and creates a composite service level request out of a list of service level requests. New requests can be put into the data structure, and they can be canceled using their unique id.

4.2 Expression Engine

The evaluation of a context expression can be triggered as a result of two types of events:

- One of the expression's context entities changes value
- The evaluation is triggered by the *scheduler*

The first type is implemented as a simple *observer* pattern. When an expression is added, a list is made of all the entities it depends on, and the root of the expression tree is subscribed to all those entities. Whenever a new reading comes in for any of those entities, re-evaluation of the entire expression tree is requested asynchronously.

The second type is implemented by using Android's AlarmManager. The alarm manager makes sure that the device wakes up whenever an alarm is scheduled, if it is in sleep mode. The evaluation scheduling system also uses asynchronous evaluation requests.

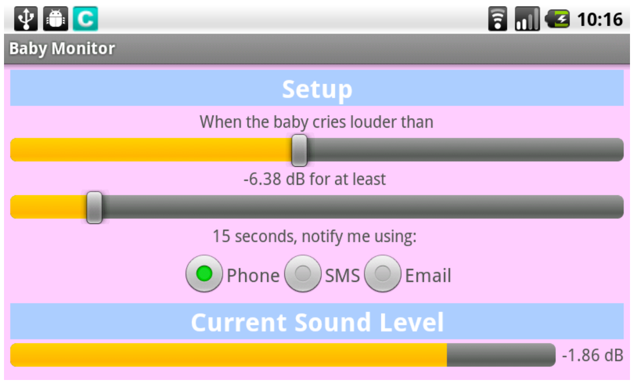
The asynchronous approach was chosen because it prevents blocking in situations where one event triggers the evaluation of an expression that is already being evaluated at that moment.

We use a top down approach for evaluating tree expressions. That is, whenever any of the Context Entities that any of the tree's leaf nodes depend on change, the whole tree is re-evaluated in a top-down order. We use this approach because it is relatively simple to implement short-circuit evaluation. When evaluating an AND node, for example, the right operand only has to be evaluated if the left operand is true. Even though "A AND B" is the same as "B AND A" from a logical point of view, the programmer can optimize the energy- and time consumption of the expression by considering the order of the operands.

5. EXAMPLE: BABY MONITOR

To illustrate the use of ContextDroid we created a context aware application, which allows a smartphone to be used as a baby monitor. The phone will monitor the sound level and when a certain threshold is passed the application can notify another phone by calling, texting or even mailing a small video clip (see Figure 2).

The code excerpt in Figure 3 shows the lines in the application that deal with the context.



When the baby cries louder than

-6.38 dB for at least

15 seconds, notify me using:

[✓ Phone] SMS Email

-1.86 dB

Figure 2: Screenshot of the Baby Monitor App

```
manager = new ContextManager(this, new ContextManagerListener() {
  // connected with service
  public void onConnected() {
    // look at the minimum sound level over the last time period (15 s)
    selector = new EvaluatorInputSelector("sound.level_rms", MINIMUM,
      timeThreshold);
    // check the sound level against this threshold (−6.38 dB)
    parameters.putDouble("value", soundLevelTreshold);
    // and evaluate whether it's greater than or equal to it
    condition = new ContextCondition(">=", parameters, selector);
    // add the condition to the manager
    manager.addContextExpression(condition, "baby", serviceLevelRequest);
    // and if it evaluates to true, notify another phone
    manager.registerContextListener("baby", new ContextListener() {
      public void onTrue(String expressionId) {
        notifyAnotherPhone();
      }
    });
  }
});
```

Figure 3: Code Example of Baby Monitor App

6. RELATED WORK

A project similar to ContextDroid is Context Weaver [1], which was developed at IBM in 2004. It is a platform that simplifies writing of context-aware applications. It lets applications access context information through a simple, uniform interface. Applications access data not by naming the provider of the data, but by describing the kind of data they need, after which the system will respond with a suitable provider. An important aspect of Context Weaver is that when a provider fails, Context Weaver automatically tries to find another provider of the same kind of data.

Context Weaver only considers current values of context, whereas ContextDroid includes historic information and adds expiration times to values, which results in more accurate context data. Furthermore, ContextDroid has specifically been designed for mobile platforms and takes energy usage into account, while to our knowledge there are no reports of Context Weaver running on mobile platforms.

WildCAT [2] is a Java toolkit/framework whose goal is the same as ContextDroid's: to ease the creation of context-aware applications for application-programmers. WildCAT too offers an API for programmers to access context information both synchronously and asynchronously. WildCAT uses a string based expression model, which is largely equivalent to ContextDroid's expression model.

WildCAT does not offer any means of service level management such as ContextDroid does. And although WildCAT is written in Java and in theory could be easily ported to mobile devices, it has not been designed especially for mobile platforms and for instance the lack of service level management makes it less suitable for mobile platforms, since efficiently handling the devices resources is of key importance on mobile platforms.

FRAP [4] is another context framework targeted at the construction of pervasive (multi-player) games. In FRAP, a central server keeps track of all context information of the clients, which have to be connected to the server. FRAP uses *WildCAT2* [2] to store context information and thus is also not appropriate for mobile platforms.

7. FUTURE WORK

Our future work with the framework will involve further evaluating and optimizing the energy consumption of the framework. We also intend to look further at usability and extensibility through the construction of more context aware applications. We will also add support for distributed context expressions which run over multiple devices in order to enable distributed context applications. For instance a user may request to be notified to initiate a call when both they and their partner are not in meetings. Finally, we intend to explore context policy enforcement with our framework.

8. CONCLUSIONS

In this paper we have presented ContextDroid, a framework that eases the development of context aware applications for smartphones. We designed and implemented ContextDroid based on the requirements for a context framework targeted at smartphones: usable, efficient, extensible and portable.

ContextDroid offers a simple, uniform and intuitive way for applications to register context expressions. Due to the centralized setup ContextDroid integrates multiple context expressions and computes a composite service level, such that multiple application requirements are met with the lowest pressure on the device's resources.

We have evaluated the ContextDroid framework with a real world smartphone application.

9. REFERENCES

[1] N. Cohen et al. Building Context-Aware Applications with Context Weaver. *IBM Research Division, TJ Watson Research Center*, 2004.

[2] P.-C. David and T. Ledoux. Wildcat: a generic framework for context-aware applications. In *MPAC '05: Proceedings of the 3rd international workshop on Middleware for pervasive and ad-hoc computing*, pages 1–7, 2005.

[3] R. Kemp, N. Palmer, T. Kielmann, and H. Bal. Opportunistic Communication for Multiplayer Mobile Gaming: Lessons Learned from PhotoShoot. In *MobiOpp '10: Proceedings of the Second International Workshop on Mobile Opportunistic Networking*, pages 182–184, 2010.

[4] J.-P. Tutzschke and O. Zukunft. Frap: a framework for pervasive games. In *EICS '09: Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems*, pages 133–142, 2009.

An Integrated Monitoring System for Smartphones

Christopher Miller
University of Notre Dame
miller.444@nd.edu

Sarah Chasins
Swarthmore College
schasil@swarthmore.edu

Carolyn Farris
University of Portland
farris11@up.edu

Justin Varner
Penn State University
jthev05@gmail.com

Curtis Carmony
Bard College
curtis.carmony@gmail.com

Christian Poellabauer
University of Notre Dame
cpoellab@cse.nd.edu

ABSTRACT
Much work has been done in the area of monitoring on traditional systems, such as servers, workstations and laptops. User and application behavior has also been studied on a wide range of platforms. Recently, smartphones have seen a dramatic increase in availability and adoption. New monitoring tools are needed to handle the unique demands of these mobile devices, such as minimal energy usage and cellular network activity, and the unique opportunities they provide, such as incorporating contextual information. Smartphones include a wide range of sensors which can be used to provide insights about the context of the activities being monitored. Individuals also use their mobile phones in a much different manner than traditional systems, and these differences have not been fully explored. With a better understanding of how these devices are used, and how common usage patterns impact system performance, we can improve upon system and application design. In this paper, we introduce an integrated monitoring framework for mobile phones which incorporates sensor, system and user activity. Our expectation is that integrated monitoring solutions will provide the foundation for various new solutions.

1. INTRODUCTION
The rapid increase in smartphones over the past few years is having a significant impact on the primary computing interface that many individuals use. As highly capable mobile devices become more prevalent, more activities are being pushed to these mobile devices, rather than using more robust systems. Much work has been done to study usage on workstations and laptops [6, 1, 12, 7, 2, 4, 9], but typical usage on smartphones can differ significantly. Smartphones are natural candidates for more context-aware applications, since they typically contain multiple sensors, such as GPS and light sensors, which can provide information about the users environment. Understanding how users interact with these devices can help developers optimize applications and system level components to improve performance and make devices more efficient.

While previous work has focused on system and performance monitoring on traditional devices, and on sensor and user activity on mobile devices, we feel that utilizing an integrated cross-layer monitoring tool which incorporates all information provides an opportunity for novel and interesting discoveries. For example, better understanding of the relationship between user activity and sensor readings will allow for analysis of complex socio-technical interdependencies (e.g., how technology affects human activity and interaction and how technology can be improved to better support next-generation social networks). Another example would be to study the interaction between sensor readings or user activity and operating system activity. This could provide new insights into the performance impacts of common usage patterns and applications, which can be used for the optimization of operating systems or applications. We aim to provide an integrated monitoring tool which incorporates the three layers of activity: sensor, system, and user.

We design our monitoring framework for the Android mobile operating system. There are several advantages to working with this operating system. Firstly, Android is an open operating system, so much of the source code is easily accessible. The openness of the platform has also resulted in a large community of developers. Android is based on a standard Linux kernel, which allows us to incorporate methods utilized in Linux. Lastly, Android is a popular and growing platform. It currently sells around 200,000 devices per day, meaning there is a very large and rapidly growing user-base for this platform. Using a combination of kernel patches and kernel modules, we develop an abstracted interface layer for monitoring. In this paper we introduce NDroid, an integrated monitoring solution which provides a simple API for research and development.

2. RELATED WORK
Significant work has been done in the areas of monitoring and modeling on traditional systems. Tools such as PerfMon [7] and PAPI [6] provide low level system performance monitoring. Ganglia [9] and SuperMon [12] provide monitoring tools for clusters and distributed systems. There has been previous work on modeling user or system behavior [4, 1, 2] as well, which is used to improve system performance or design. Some of these same concepts and methods can be utilized in monitoring smartphones, but smartphones present a greater importance on energy conservation, and provide additional contextual information that can be incorporated into the monitoring. Some early work in modeling user activity on smartphones has been completed by Falaki et al. [8].

Previous work has attempted to extract contextual information about users based on embedded or wearable sensors, such as the Mobile Sensing Platform [5]. Recent work has extended this concept to utilize the sensors available in smartphones, since these are becoming ubiquitous wearable sensing devices. Reddy et al. used smartphones to determine the transportation mode of an individual [14], while the CenceMe project attempted to decipher multiple contextual properties such as user activity and location, using a variety of sensors and learning algorithms [11]. More recent work has also attempted to use sensor data from smartphones and user interaction with applications to model human behavior [3, 13, 10, 11]. These studies can provide interesting discoveries which can aid in future system and application design.

3. MONITORING ON MOBILE PHONES

While system monitoring on standard systems has been extensively researched, little work has been done in the area of monitoring on small mobile devices. Mobile phones provide some unique opportunities and challenges for monitoring. Since mobile phones have limited resources in comparison to standard systems, efficiency is imperative. Mobile phones run on batteries, therefore monitoring services should have minimal impact on power usage, so that lifetime of devices is not diminished. Also, since mobile phones have relatively limited processing power and memory, the computational impact on systems should be minimized. Smartphones offer several unique opportunities as well. Smartphones incorporate various sensors, such as GPS, accelerometers, orientation sensors, and ambient light sensors. These sensors can provide context information about the user and their environment. These aspects of usage could not be as easily integrated with system monitoring on standard systems, and could provide new understandings of how systems are used. Also, smartphones are utilized differently than standard systems. This provides an opportunity to understand unique usage patterns on a new platform.

Early efforts to model smartphone usage have shown that there is a great diversity among how users interact with these devices [8]. These findings indicate that optimization of applications may best be achieved using dynamic methods which depend on the user or usage scenario. Developers could incorporate monitoring services into their applications to obtain information which may be useful for optimization. This approach would not be ideal, however, as many developers would likely be incorporating similar monitoring features into their applications, resulting in inefficiencies in the overall system. This is particularly significant in smartphones, where resources are limited, so duplicated monitoring services can be expensive. It can also be difficult to monitor some system level information, especially if it is hardware dependent. Providing a single, simple to interface, abstracted monitoring layer can greatly improve efficiency of monitoring and make it much easier for developers to incorporate monitoring tools into their applications, which could lead to novel improvements. This monitoring layer could handle requests from all applications, and provide a single interface to monitoring tools, which would eliminate inefficiencies due to duplication of monitoring services. As an example, if three applications needed to monitor the CPU load at frequencies of 10 Hz, 20 Hz, and 100 Hz, respectively, the monitoring layer could service all three of these requests using a single 100 Hz monitoring service.

We developed the NDroid monitoring system to be inclusive of all metrics which may be useful to developers or researchers. The features which are monitored in the system may be considered to fall in three categories: sensors, system, and user activity. Sensors include any metrics provided by readings from available sensors on the phone. This feature list will vary somewhat based on the device, but most smartphones have a fairly standard group of sensors, such as GPS and accelerometers. System features include any metrics pertaining to the available resources of the system, and any activities managed by the operating system or kernel. The system resources would include processing power, battery power, memory capacity, and network capacity. System activity would include read/write operations, network actions such as sending a packet, and processor actions such as context switches. All system features are monitored at the kernel level. This is currently accomplished using a patched kernel and kernel modules, but preferably these features would later be pushed upstream to the main Android kernel so that it will be available to all users. To provide an efficient implementation which is consistent with current kernel practices, monitoring features are implemented utilizing the Linux notifier toolchain, which provides a publish/subscribe method of notifications. This method will also allow the monitoring system to customize what is currently being monitored based on application needs, thereby avoiding unnecessary use of system resources. To provide a complete monitoring solution, NDroid will also incorporate user activity monitoring. This includes monitoring of activities such as application usage and text messaging. Integrating this information with system monitoring and sensor data provides the opportunity for stronger analysis of user activity and system performance. The following three sections provide more detail about the three categories of monitoring features, and the specific metrics which are monitored.

3.1 Sensors

Sensors can be used to determine contextual information about the device and user. This data can be used to infer user activity and environmental conditions. Combining this information with system and user activity monitoring may provide unique insights into how user interact with mobile devices. Smartphones include an increasing number of sensors, and research into how these can be used to infer activity or contextual information has grown over the past few years [14, 10]. As research in this area expands, there will only be an increase in the contextual information that can be derived from sensors. A list of the implemented and planned monitoring features for sensors is shown in Table 1.

Table 1: Sensor monitoring features

Activity	Feature	Description
GPS	gps	GPS sensor reading
Magnetometer	magneto	Magnetometer sensor reading
Accelerometer	accelx accely accelz	Accelerometer sensor reading for x, y, and z axis
Orientation	azimuth pitch roll	Orientation sensor reading for azimuth, pitch, and roll
Proximity	proximity	Proximity sensor reading
Light	light	Ambient light sensor reading

3.2 System

System monitoring encompasses most of what would be considered typical performance and system monitoring on traditional systems. In includes information about system resources and operating system activity. System resources encompass the resources available to a system which directly impact user experience. These resources include the processor, memory, and network resources, as well as vital system resources such as the battery. A list of the implemented and planned monitoring features for system resources is shown in Table 2.

Table 2: System resource monitoring features

Activity	Feature	Description
CPU	cpufreq	Current CPU frequency
CPU	cpuload	CPU load
Memory	memavail	Available memory
Cache	cache	Cache in use
Wifi Network	wifibw	Bandwidth of 802.11 interface
Wifi	wifisig	Signal strength of 802.11 connection
Cellular Network	cellbw	Bandwidth of cellular data interface
Cellular Network	cellsig	Signal strength of cellular connection
Battery	battlevel	Remaining battery level (as percentage)
Battery	battcurr	Active current usage of system (in mA)
Battery	battcap	Capacity of battery

Monitoring system resources can provide valuable information for a number of applications. It may be used to optimize the performance of a system by altering the function of the system based on observed usage or available resources, as is done in a userspace frequency scaling application. Applications can use this information to alter their behavior in order to avoid negatively impacting the system or user experience. System resource monitoring may be used to model the impact of applications on a system, and can be a tool for optimizing applications. It may also be used to model user behavior and activity, to determine methods to optimize the system based on usage.

System activities include any action of interest which may be taken by the system, typically utilizing system resources. When integrated with system resource monitoring, these features can be used to study how activities impact system resource usage. When integrated with user activity monitoring, these features can be used to study how user activity impacts the system. These features may also be used for modeling of applications. It is possible that each application has a fingerprint, which can be determined based on observed system activity. Modeling system activity for different applications could provide useful insight to how the system is being used, and what actions may be expected in the future. This could allow developers and researchers an opportunity to optimize system performance based on expected future needs. This information can also be useful for applications, which may need to know the state of some system components. A list of the implemented and planned monitoring features for system activity is shown in Table 3.

Table 3: System activity monitoring features

Activity	Feature	Description
CPU	context	Context switch on the processor
I/O	memread memwrite	Read/Write internal memory
I/O	sdread sdwrite	Read/Write sdcard or external storage
Interface Up / Down	netup netdown	802.11 interface up/down
Interface Up / Down	cellup celldown	Cellular interface up/down
Interface Up / Down	blueup bluedown	Bluetooth interface up/down
Interface Up / Down	gpsup gpsdown	GPS interface up/down
Devices	blueconn bluedisc	Bluetooth device connect/disconnect
Network	nettrx	802.11 packet transmit
Network	netrecv	802.11 packet receive
Network	celltrx	Cellular network packet transmit
Network	cellrecv	Cellular network packet receive

3.3 User activity

User activities encompass activities observed at the application layer. This information can be very useful for understanding typical usage patterns of smartphones. Mobile phones provide a unique environment for system resource usage, which has not been fully explored. Modeling user behavior and developing a better understanding of how these devices are used can be instrumental to efforts to optimize system performance. When combined with system monitoring and sensor data, this data can also provide information about how applications impact system activity and resource usage. This can be a great tool for analyzing and improving application performance. A list of the implemented and planned monitoring features for user activity is shown in Table 4.

Table 4: User activity monitoring features

Activity	Feature	Description
Application	appopen appclose	Open / Close of application by user
Cellular activity	callsnd callrec	Make / Receive a call on cellular network
Cellular activity	textsnd textrec	Send / Receive a text on cellular network
Email	emailsnd emailrec	Send / Receive an email
Screen	screen	On/Off state of screen

4. MONITORING API

The primary purpose of this monitoring tool is to facilitate development and research for smartphones and mobile devices. To do this, we aim to provide a simple and easy to interface API, which will allow developers and researchers to easily access the desired metrics from the system and still provide a robust tool which allows them to customize the monitoring metrics to their specific needs. To accomplish this, we use a kernel module, which serves as a central manager for the monitoring processes, and a central point of communication for the monitoring system and the developer. The NDroid module will expose an API to the developer which will allow them to request the features they want monitored, and the properties for each monitoring activity. The module will then communicate with other modules, and with the kernel, to initiate only those monitoring services which are needed. This will allow the system to provide a very robust monitoring tool without having any greater impact on the system than is necessary. The NDroid API may be accessed by multiple applications, the module will manage which features are needed by each requester, and provide each only the information that is requested.

There are three primary types of requests that can be made to NDroid: instantaneous reading, register a continuous monitor, and register a notification monitor. The instantaneous reading request is simply a request for the current value of a particular feature that can be monitored. For instance, this can be a request for the current load on the CPU. This a single instance request, for cases when a continuous monitor is not necessary. The format for such a request in the NDroid API is:

```
instant_value(u16 FEATURE, (void *) VALUE)
```

`FEATURE` indicates the desired metric to be measured or read, and `VALUE` is a pointer to the location where the value of the metric should be stored.

For metrics that need to be continuously monitored over time, the developer may register a continuous monitor of a supported feature. They can do so by indicating the feature they want to monitor, as well as the frequency of readings or measurements. They will also indicate a callback function for this monitor. This callback function will be used in a similar fashion as the Linux notifier toolchain. The NDroid module will call the function at the requested frequency, including in the call a single argument which is the value of the requested feature. The developer will need to provide a function which handles this value to process it as desired. This method is used to avoid unnecessary reads/writes to the system or to external storage. This helps keep the monitoring tool lightweight by passing the value directly to the requester(s) rather than writing to the proc directory or to a file. The format for such a request in the NDroid API is:

```
register_monitor(u16 FEATURE, time_t FREQUENCY,
(void *) FUNCTION)
```

The NDroid monitoring system can also be used to continuously monitor features but only issue callbacks when certain conditions are met. These monitors will be referred to as notifiers. The monitoring module will continuously monitor these features at the requested frequency, but will only initiate a call to the callback function when the specified criteria are met. The format for such a request is similar to the monitor request, but with the additional information to specify the notification conditions. These conditions may be absolute qualifiers, such as notify when the feature value rises above a certain threshold, or relative qualifiers, such as notify when the feature value changes by a certain threshold. The supported condition types are shown in Table 5. The format for such a request in the NDroid API is:

```
register_notifier(u16 FEATURE, time_t FREQUENCY,
u16 CONDITION, u16 VALUE, (void *) FUNCTION)
```

Table 5: Supported condition types for registered notifier monitor

Condition	Description
MINTHRESH	Notify when metric falls below a minimum threshold
MAXTHRESH	Notify when metric goes above a maximum threshold
CHANGE	Notify anytime there is a change in metric value
UPTHRESH	Notify when metric rises a specified threshold
DOWNTHRESH	Notify when metric falls a specified threshold
ABSOLTHRESH	Notify when metric rises/falls an absolute threshold

5. FUTURE WORK AND DISCUSSION

We are currently implementing the NDroid monitoring tool, and plan to fully implement all monitoring features described above, as well as any additional features which are found to provide useful information. The API module will be implemented to manage each of these metrics, and to expose the described interface to developers for ease of integration. We will also look into alternate methods of monitoring frequency. Rather than leaving frequency of monitoring decisions to users or developers, an alternate option would seek to provide an optimal frequency, such as the Markov-optimal sensing policy proposed in [15]. Following completion of implementation, we will extensively test all features of the monitoring tool to measure its impact on energy usage and processing latency. It is always important to minimize impact on a system when monitoring, but given the limited resources of smartphones, it is especially critical. The monitoring tool has been designed with this in mind, and should be fully tested to ensure that it meets these expectations. Impact to both battery lifetime and system responsiveness should be negligible to users.

Future work will also need to include an examination of the privacy consequences of this tool. The tool will of course be designed to only capture state information which could be useful to understanding usage, such as capturing a text message send event, without capturing the text in the message. Despite this, the tool will have the ability to generate a detailed log of what applications the user used, at what times, and under what environmental conditions (based on sensor readings). This could be considered a privacy control issue, even though specifics of the application use are not captured. This may not be an issue for research which is conducted with a controlled group of active participants. However, if this tool should become an integrated part of Android to support simple access to system resource data by application developers, then these privacy issues will need to be addressed.

There is still much to be learned about how individuals utilize their smart and multimedia phones. Our current knowledge base is rooted primarily in common usage on traditional systems. Usage patterns on mobile phones may, and likely do, differ significantly than usage on traditional systems. Understanding how these devices are utilized can provide valuable information to optimize both hardware and software design, and improve performance and utility. This monitoring tool will be used to capture data on application usage and typical user activity. We hope to use this data to develop models which will provide a better understanding of how applications impact the system and how users utilize smartphone devices. These models could provide the insight needed to improve system performance, or the performance of applications.

6. ADDITIONAL AUTHORS

Additional authors: Aaron Striegel (University of Notre Dame, email: `striegel@nd.edu`)

7. REFERENCES

[1] L. A. Barroso, K. Gharachorloo, and E. Bugnion. Memory system characterization of commercial workloads. In *25th Annual International Symposium on Computer Architecture*, pages 3–14, 1998.

[2] L. Benini, A. Bogliolo, and G. D. Micheli. A survey of design techniques for system-level dynamic power management. *IEEE Trans. VLSI Syst*, 8(3):299–316, 2000.

[3] F. Bentley and C. Metcalf. The use of mobile social presence. *IEEE Pervasive Computing*, 8(4):35–41, 2009.

[4] S. Charbonnier, C. Garcia-Beltan, C. Cadet, and S. Gentil. Trends extraction and analysis for complex system monitoring and decision support. *Eng. Appl. of AI*, 18(1):21–36, 2005.

[5] T. Choudhury, G. Borriello, S. Consolvo, D. Haehnel, B. Harrison, B. Hemingway, J. Hightower, et al. The mobile sensing platform: An embedded activity recognition system. *IEEE Pervasive Computing*, pages 32–41, 2008.

[6] J. Dongarra, K. London, S. Moore, P. Mucci, and D. Terpstra. Using PAPI for hardware performance monitoring on Linux systems. In *Conference on Linux Clusters: The HPC Revolution*, National Center for Supercomputing Applications (NCSA), University of Illinois, Urbana, IL, June 2001.

[7] R. Enbody, K. Pellini, and W. Moore. Performance monitoring in advanced computer architecture. In *Proceedings of the 1998 workshop on Computer architecture education*, page 17. ACM, 1998.

[8] H. Falaki, R. Mahajan, S. Kandula, D. Lymberopoulos, R. Govindan, and D. Estrin. Diversity in smartphone usage. In *MobiSys*, pages 179–194. ACM, 2010.

[9] M. L. Massie, B. N. Chun, and D. E. Culler. The ganglia distributed monitoring system: design, implementation, and experience. *Parallel Computing*, 30(5-6):817–840, 2004.

[10] E. Miluzzo, C. Cornelius, A. Ramaswamy, T. Choudhury, Z. Liu, and A. T. Campbell. Darwin phones: the evolution of sensing and inference on mobile phones. In S. Banerjee, S. Keshav, and A. Wolman, editors, *MobiSys*, pages 5–20. ACM, 2010.

[11] E. Miluzzo, N. Lane, K. Fodor, R. Peterson, H. Lu, M. Musolesi, S. Eisenman, X. Zheng, and A. Campbell. Sensing meets mobile social networks: the design, implementation and evaluation of the cenceme application. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 337–350. ACM, 2008.

[12] R. G. Minnich. Supermon: High-performance monitoring for Linux clusters. In *Proceedings of the 5th Annual Linux Showcase and Conference*, Nov. 2001.

[13] D. Peebles, H. Lu, N. Lane, T. Choudhury, and A. Campbell. Community-Guided Learning: Exploiting Mobile Sensor Users to Model Human Behavior. 2010.

[14] S. Reddy, J. Burke, D. Estrin, M. Hansen, and M. Srivastava. Determining transportation mode on mobile phones. 2008.

[15] Y. Wang, B. Krishnamachari, Q. Zhao, and M. Annavaram. Markov-optimal sensing policy for user state estimation in mobile devices. In *IPSN*, pages 268–278. ACM, 2010.

BSMX - A Prototype Implementation for Distributed Aggregation of Sensor Data

Sascha Schnaufer
Computer Science IV
University of Mannheim
schnaufer@informatik.uni-mannheim.de

Stephan Kopf
Computer Science IV
University of Mannheim
kopf@informatik.uni-mannheim.de

Wolfgang Effelsberg
Computer Science IV
University of Mannheim
effelsberg@informatik.uni-mannheim.de

ABSTRACT
Beacon-based Short Message eXchange (BSMX) is a system to exchange small-sized messages between unassociated WLAN devices like smartphones or access points. In this paper, we describe our proof of concept implementation of BSMX for Linux and the Android operating system. Furthermore, we introduce a novel probabilistic data structure that BSMX utilizes to provide application developers methods for distributed data aggregation.

1. INTRODUCTION
Vehicular Ad-Hoc Networks (VANETs) are a special class of Mobile Ad-Hoc Networks (MANETs) where road vehicles with WLAN equipment form a network without additional infrastructure. In such a network each vehicle can communicate directly with all other vehicles in radio range. Typical applications for such a network try to increase the driver's safety and convenience by exchanging sensor values. For instance, SOTIS [8] and TrafficView [5] exchange information like speed and position among vehicles in order to enable users to access the current traffic conditions. Furthermore, [1] propose to equip ticket machines with WLAN so that they can inform vehicles about the capacity utilization of their parking lots. These approaches have in common that vehicles automatically aggregate received information and exchange these aggregates among each other. If a received aggregate is outdated or the distance between the vehicle and the position of the data origination is too large, the information is dropped and the dissemination is stopped. Instead of querying special information, the system works as a best effort service and automatically exchanges aggregates about the situation in the proximity of a vehicle. Some approaches also propose a hierarchical aggregation system. For instance, the authors in [1] subdivide the plane into a grid and utilize a quad tree mechanism to aggregate the utilization of parking lots. This system enables the driver to receive information about available parking lots in the proximity, but also about the situation in other districts.

The main purpose of VANETs is to increase the security by warning other drivers of dangerous situations like an emergency braking. It is obvious that due to the high risk of misuse, such a network does not use an open architecture that can be utilized to develop novel applications by everyone. The main contribution of this paper is the presentation of our technical solution to build such systems based on already deployed IEEE 802.11 devices like access points and smartphones that primarily have another intended use.

Phonesense'10 at ACM SenSys'10, Nov. 2, 2010, Zurich, Switzerland.

The remainder of this paper is structured as follows: The next section analyzes the mechanisms to exchange data packets between mobile devices like smartphones. Section 3 describes our novel approach called BSMX which allows the easy data exchange in a 1-hop neighborhood. A proof of concept implementation is described in Section 4. Section 5 explains an existing probabilistic data structure that can be used for the in-network aggregation of sensor values. Our novel data structure that is used in the BSMX system is described in Section 6. Section 7 illustrates the results of our simulation study. Finally, Section 8 summarizes the paper.

2. WIRELESS COMMUNICATION
Several mainstream wireless communication technologies for handheld mobile devices like smartphones are available on the market today. However, there is no established and easy to use method available to exchange packets between such devices without using any infrastructure. On the one hand, smartphones are equipped with technologies like UMTS and GSM, both using an area-wide infrastructure which is controlled by telecommunication companies that offer the access to the phone network and to the internet via their infrastructure. On the other hand, smartphones are typically also equipped with Wireless LAN (WLAN) and Bluetooth. The telecommunication companies have no incentive that devices of their customers communicate directly with each other without the usage of the companies' infrastructure and outside of their control. The situation of the second group of technologies differs clearly, because in the most cases WLAN and Bluetooth are self-governed by the device owners.

Most smartphones are equipped with a Bluetooth device of class 2 or class 3 and have a very limited radio range. Two devices need to be paired to communicate with each other. The pairing process is typically triggered automatically the first time a device receives a connection request. After both users have entered the identical pin the two devices can exchange files or contact information. Multicast or broadcast communication is only available if the devices operate in a Bluetooth ad-hoc network which is called piconet. However, the Bluetooth stacks of smartphones usually do not support the required profiles to operate in ad-hoc mode. Therefore, we focus the further discussion on IEEE 802.11 which is intended as a general replacement for wired networks and allows a more flexible configuration.

Although supported by IEEE 802.11, the ad-hoc mode that allows a direct communication from one device to other devices in radio range is rarely used in practice. We assume that the complex configuration of ad-hoc networks is the main reason for the current situation. IEEE 802.11 client devices can start a search over all radio channels to find access points and ad-hoc networks in radio range. If the user wants to join a discovered network the device can adopt the required network properties like SSID, radio channel and encryption method from the selected network. However, to create a new ad-hoc network the user has to configure the device manually. Furthermore, the network requires a mechanism to assign unique IP addresses to all devices, and if multi-hop communication is wanted a special ad-hoc routing protocol is required. Another issue is that many devices cannot connect to an access point for Internet access and communicate at the same time with other ad-hoc devices. We are confident that most of these configuration problems could be solved with the adoption and extension of existing technologies, but the tools the devices provide today are not sufficient or far too complex. Another reason why ad-hoc communication did not find the way to the market yet is that no unique feature or killer application is available which requires it. Our conclusion is that the already widely-used mobile Internet connections via UMTS are sufficient to enable end-to-end communication between mobile devices. However, we assume that an easy to use 1-hop data exchange mechanism can generate a significant user benefit and enable new types of applications.

3. BEACON-BASED SHORT MESSAGE EXCHANGE

The IEEE 802.11 standard defines a set of procedures to create, join and maintain WLAN networks. These mechanisms require additional packets that are not forwarded to the operating system. Furthermore, the standard allows manufacturer-specific components in most of these management packets. We have developed an IEEE 802.11-compliant extension that allows user space applications to add short messages to these management layer packets. Moreover, we also developed a mechanism that forwards this user data to the related application. A major advantage of our approach is that it does neither require a common SSID nor negotiate encryption settings or routing layer configurations. The idea is that WLAN devices can operate without changing their network and security configurations. We call this novel communication method *Beacon-based Short Message eXchange* (BSMX).

Access points and ad-hoc network nodes send unencrypted beacon packets periodically, typically every 100 ms. Our approach can be utilized to add small-sized messages to these beacon packets which are sent independently from the current network load, anyway. This exchange method leads to a heterogeneous radio channel configuration. However, the distance between the IEEE 802.11b/g channels is 5 MHz only, and the used channel width is 22 MHz. This overlap and the used encoding technique allow the successful decoding of some packets which are transmitted on adjacent channels by using a technique called overhearing. Network devices drop such packets by default, but our BSMX system uses this channel overlap to monitor a part of the frequency band without changing the radio channel of the device. In a comprehensive measurement study we analyzed the expected connectivity between indoor access points and a mobile device on the street in an inner city environment [7]. The average reception rate is 53% for packets that are sent on the same channel, 35% for packets that are sent on adjacent channels and 4% for packets with a distance of two channels. The low rate of 53% is caused by the fact that most access points should only cover indoor areas, whereas the measurement was performed on the street.

One drawback of this mechanism is the fact that mobile devices like smartphones typically run in client mode and hence they do not send beacon packets continuously. Thus, a mobile device running in client mode can receive messages from access points which operate on the same radio channel but cannot send messages back or even communicate with other client devices. Therefore, the extended version of our approach utilizes the active scan procedure which is defined in the IEEE 802.11 standard to discover access points in the proximity. During such an active scan the device passes through all radio channels to send probe request packets that are answered from receiving access points by returning a probe response packet. The BSMX system can add messages to both probe request and probe response packets and can this way exchange small-sized messages between unassociated WLAN devices that do not operate on the same radio channel. This approach has the advantage that most devices can conduct an active scan while they are connected to an access point and smoothly resume the connection after the scan. The active scan mechanism can also be utilized to exchange messages between devices that currently are not connected to an access point.

4. IMPLEMENTATION

The chance that BSMX can find a way to the consumer market highly depends on the complexity of its appropriation. Therefore, a main design goal of the BSMX system is to minimize required changes of existing software components and work principles. This consideration is the reason why BSMX utilizes the so called tagged parameter mechanism of the IEEE 802.11 standard to extend already existing management packets. This strategy has two important advantages: First, devices without BSMX support just ignore unknown tags and discard the additional data without any drawback or failure. The second advantage is that the extension of existing device drivers is comparatively easily and can be done by including the source code of our extension in less than hundred lines of code.

Figure 1 shows the system architecture. The BSMX extension is linked to the device driver and is running in kernel space. We use Netlink (RFC3549) as communication method between the extension and user space applications. The advantage of Netlink compared to other communication methods like ioctls is that it allows bidirectional communication and implements a multicast mechanism. The former one is necessary to forward received messages to user space without polling. We have implemented the extension for the TNETW driver that is used by the majority of Android devices and also for the open source driver MadWifi which supports Atheros based chipsets.

The BSMX header specifies the message type as integer value but does not implement an addressing schema. We

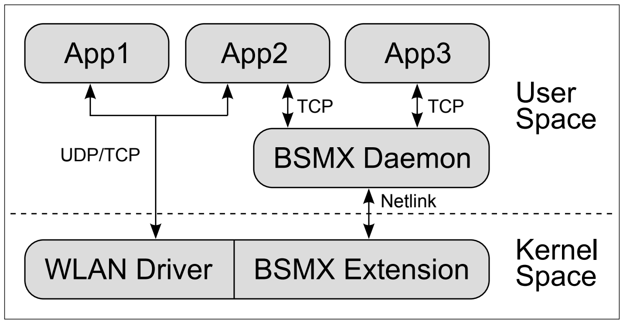


Figure 1: BSMX System Architecture

assume that most applications decide on the receiver's side whether they are a recipient. With respect to the broadcast characteristics of the ether every transmitted message can be received by every device in radio range. A new aspect is that we move the decision whether a packet should be dropped from the operating system to the application. However, it is possible that more than one application is interested in receiving this message. For this purpose we developed a central daemon that implements the publish/subscribe pattern and enables applications to subscribe the reception of messages of a specified type via TCP. Applications can also utilize the daemon to send messages over WLAN or to other local applications. Furthermore, the daemon maintains a neighbor table that can be accessed by all applications. The central instance is also required to check the security settings and deny unauthorized access to the WLAN device. Additionally, the daemon provides a standardized interface to several probabilistic data structures that can be utilized by application developers.

The intention of this approach is to provide a complete set of tools to create new types of applications. For instance, an application can aggregate the sensor values of several devises in the proximity before transferring the result to a central server for further processing. This procedure would help protecting the privacy of the application users. We also assume that for many applications it would be sufficient to know that two devices are close to each other without knowing where they are.

5. FLAJOLET-MARTIN SKETCHES

A Flajolet-Martin sketch is a data structure for probabilistic counting of distinct elements [3]. In this context an *element* represents everything for that a hash value can be calculated, e.g., a text string or a file. Please consider that the data structure does not store the element itself. The advantage is that the required storage size of the sketch only depends on the selected configuration parameters and does not depend on the number of inserted elements. A higher storage size leads to a better approximation of the number of distinct inserted elements. Two parameters influence the required storage size. M defines the accuracy and L the number of distinct elements that can be inserted. For instance, $M = 128$ and $L = 16$ leads to a storage size of 256 Bytes ($128 \cdot \frac{16}{8}$). The sketch of this example can count about 530000 distinct elements and provides a standard error of approximately 7% ($0.78/\sqrt{M}$).

An application X can add the integer value Y to a sketch by inserting Y different elements. For instance, the application can use the hash values of the strings "$X : 1$" ... "$X : Y$". In this scenario, the number of distinct values of the sketch is equivalent to the sum of all inserted integer values. For example, if two applications add values of $Y_1 = 25$ and $Y_2 = 50$, the distinct number of inserted elements is 75. The calculation of the average requires a sketch that stores the sum of all integer values and one that stores the number of inserted integer values. In the example of the parking lot application named in the introduction, the capacity utilization can be calculated by using two sketches. One contains the number of available parking lots and the other one stores the total number of parking lots.

The insertion process is deterministic and completely independent from the current state of the sketch. If one element is inserted several times, the identical bit is set to one each time. Hence, sketches are duplicate-insensitive and the order of insert operations does not affect the estimation process. These probabilities allow that sketches can be merged by a simple bit-wise OR operation. The merged sketch can be used to estimate the total number of distinct elements that are added to any of the source sketches. This behavior and the compact storage size are useful for the distributed calculation of aggregation functions like COUNT, SUM and AVG inside VANETs or sensor networks [2]. [6] introduce a compression schema for sketches and [4] provides an aging strategy that removes outdated elements from a sketch.

These approaches are suitable for the in-network aggregation of sensor values. For example, a smartphone application wants to estimate the number of other devices in the 5-hop proximity that also runs an instance of the software. In this scenario every device maintains a sketch, adds the name of the device itself, and sends the data structure via broadcast to other devices in radio range. Received sketches are added to the own sketch via a bit-wise OR operation. Then the application use an aging strategy (e.g., [4]) to remove outdated entries and send the sketch again via broadcast to other devices; these steps are repeated iteratively. After several iterations each device can use the sketch to estimate the number of running instances in the 5-hop proximity. The number of hops is a setup parameter of the aging approach that increases the required storage size of a sketch.

The described procedure is very simple and robust, and does not require any knowledge about the topology. However, the same working principle can be used without sketches by exchanging a list that contains tuples with the name of a device and a time to live (TTL) counter. In this case, each device maintains such a list and adds its own name with the maximum TTL counter. The device decrements the TTL counter of all entries beside its own entry and removes entries with a TTL counter below zero before it sends the list via broadcast to its neighbors. If a device receives a list, it adds all unknown devices to its own list and updates the TTL counter of already known devices. This way each device can also estimate how many other devices in the proximity run the application. However, this approach is very limited because the size of the list will grow very fast. For instance, if every device adds its own MAC address and a TTL counter that can store four hops, every entry would require $6 \cdot 8 + 2 = 50$ bits. With respect to the typically used maximum transfer unit a list with 22 entries can be transferred without fragmentation at most. Compared to the sketch approach that can count several thousand entries, a capacity of 22 is extremely small.

A serious drawback of the probabilistic approach is that it is only suitable if the application can accept a standard error of at least 5%. In our simulations, values of $M > 256$ do no longer improve the accuracy significantly. Furthermore, the accuracy can only be achieved if more than $15 \cdot M$ elements are inserted. In other words the sketch we named before requires $128 \cdot 15 = 1920$ entries to achieve the standard error of about 7%. This behavior makes the configuration complex and depends on the number of participating devices. Another drawback is that the range of integer values that can be inserted is very limited. The number of elements that can be inserted to a sketch depends on its setup parameters. However, it is obvious that the insertion of 64-bit values exceed the available range clearly. For these reasons, we present in the next section a novel approach to calculate and exchange aggregates.

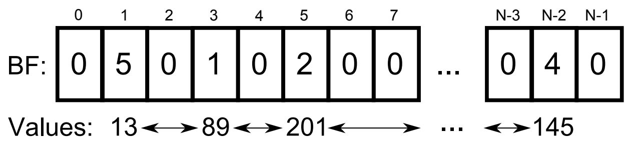
6. BLOOM FILTER MAPS

Flajolet-Martin sketches are suitable to estimate aggregation functions in large scale networks. However, in environments with a low number of participants the accuracy can decrease significantly. Another limitation is that a more complex aggregation can only be reproduced by using several sketches. This section presents a novel approach that overcomes these limitations and offers further advantages.

In the last section, we briefly described the distributed estimation of the number of participating devices by exchanging a list with device names and TTL counters. The sum or the average can be estimates by adding the sensor values to the related list entries. If we assume that it is typically suitable to use a range of $0 \ldots 255$ as sensor value and $1 \ldots 8$ as hop range, each entry needs $6 \cdot 8 + 8 + 3 = 59$ bits. It is obvious that the largest fraction of storage is used to identify the owner of the entry. The main idea of our approach is to substitute this identifier by a probabilistic replacement. Keep in mind that the hashing of the MAC address is a probabilistic method that can lead to hash collisions. However, the reduction of the storage amount achieved by hashing is not sufficient.

Bloom filters are a well-known data structure that is used to test whether an element is a member of a set. False positives are possible, but false negatives not. A Bloom filter is bit field $B = b_0 \ldots b_{N-1}$ of a length $N > 0$ and is initialized with zeros on every bit position. An element E is added by setting bit b_X with $X = hash(E)\ mod\ N$ to one. A similar procedure is also used to test whether an element was inserted before. If the tested bit is one, the element is member of the set or a collision had occurred (false positives). However, if the bit is zero, the element was definitely not inserted. The false positive rate can be reduced by utilizing more than one hash function, but we will focus on the approach with one hash function only.

It is possible to extend the array positions from a single bit to an n-bit counter. In our approach, a counter of zero means that there is no entry; otherwise the counter represents the TTL of the entry. To come back to the uncompressed list from the beginning of this section, the identifier



0 1 2 3 4 5 6 7 N-3 N-2 N-1
BF: 0 5 0 1 0 2 0 0 ... 0 4 0
Values: 13 89 201 ... 145

Figure 2: Bloom Filter Map

corresponds to the position inside the Bloom filter and the TTL counter corresponds to the counter at the related position inside the Bloom filter. The order of non-zero counters is used to maintain an additional list which contains the content of the entries, e.g., the sensor vales. We call the Bloom filter with TTL counters and value list *Bloom Filter Map* (BFMap). Two BFMaps can be merged by performing a MAX operation on every position of the related Bloom filters and arranging the corresponding values to a new list. If two positions have a non-zero counter, this process prefers the newer one. Figure 2 shows an exemplary BFMap. The upper part of the figure shows the Bloom filter with TTL counters and the lower part the list of related values.

It is obvious that several entries could be mapped to the identical position inside the Bloom filter. This represents the probabilistic part of our approach. The probability of such a collision increases with the number of already inserted entries and decreases if the size of the Bloom filter is increased. The interesting aspect here is that the enlargement of the Bloom filter only slightly increases its entropy. If the probability of each symbol of a data stream is known, the well-known arithmetic coding approach [9] can be utilized to compress the data stream entropy-optimal. A Bloom filter with T-bits TTL counters consists of the symbols $0 \ldots (2^T - 1)$. Assume that $count(X)$ is a function that returns the number of occurrences of the symbol X inside a Bloom filter, then the probability of each symbol can be calculated by $count(X)/N$. If a device receives a compressed Bloom filter, N and a list of $count(X), X \in 0 \ldots (2^T - 1)$, it can decompress the data structure. It is sufficient to use 16-bit unsigned integer values to store and transmit N and the counters.

7. EVALUATION

In [7] we estimated the connectivity of access point among each other in the inner city of Mannheim, Germany. In the following, we will use this very dense network to conduct a simulation study to estimate the accuracy and the required storage size of the BFMap approach. The simulator creates for each of the 3797 simulation nodes a BFMap and inserts a tuple based on a random integer value (node identifier) and the maximum TTL T_{max}. In the next step, the simulator adds to the BFMap of each node the values of their neighbors with a shortest path of x hops ($x \in 1 \ldots T_{max}$) with a TTL of $T - x$. This procedure emulates the TTL reduction, the exchange, and the joining process of BFMaps. Finally, the simulator compares the created BFMaps with the real situation. The comparison includes the number of neighbors (COUNT), the sum of random values (SUM) and the calculation of the average value (AVG). Furthermore, the simulator compresses each BFMap and determines its compressed size. Figure 3 shows extracts of the average results of 150 simulation runs each using 3797 nodes.

The average size of a compressed Bloom Filter entry depends

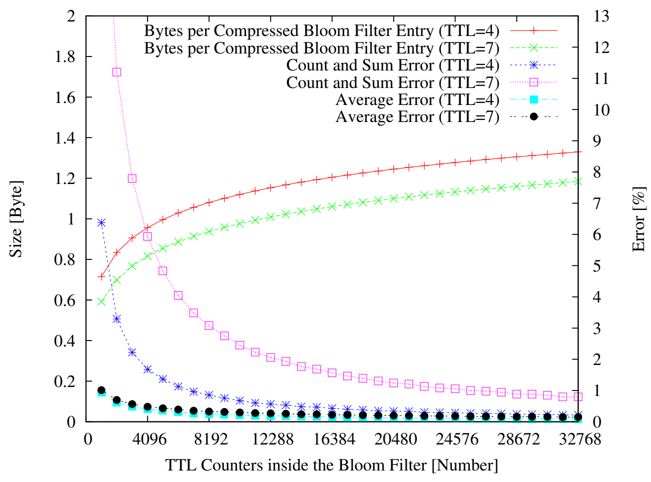


Figure 3: Simulation Results

on the number of inserted elements, the capacity N of the filter, and the range of the TTL counters. In our simulation environment the maximum TTL determines the number of neighbors inside the TTL limit and thus it also defines the number of inserted elements. Figure 3 shows the average entry size for a maximum TTL of 4 (141 entries) and 7 (520 entries) respectively. This size includes the identifier and the TTL counter of a node but not its value. Thus, if one byte per value is used, the full size of the BFMap entry is also increased by one byte. Although the compression of a higher maximum TTL requires more symbols, the average entry size of the TTL=7 example is lower compared to TTL=4. The reason for this behavior is that the required storage space of the zero TTL counters is distributed across more inserted entries.

The results show that the usage of a BFMap with 8192 TTL counters is a good trade-off between size, computational complexity, and accuracy. The average count and sum error of this setup is about 1% with 141 entries and 3% with 520 entries. An interesting aspect of the BFMap approach is that even in the case of collisions the newly inserted element only overwrites an existing entry without influencing the other elements. In other words, the number of collected samples to calculate an aggregation is reduced by one, but all remaining samples are unchanged. This effect causes the small error of the average calculation. We run simulations based on uniformly and normally distributed random values, but the differences were marginal, and therefore the figure shows the former one only.

The presented BFMap data structure is well suited for the distributed calculation of aggregation functions. The advantage is that a developer can use real values for the calculation and is not limited to basic functions like COUNT, SUM and AVG. In addition, the developer controls the size and type of the values used and decides which accuracy the application requires. For instance, the developer can use a quantization method to reduce the required size per value. Furthermore, the BFMap can be used to maintain and exchange applications states or other kind of data that is not intended for aggregation purposes. Contrary to Flajolet-Martin sketches BFMaps do not require a minimum of inserted elements to work properly. However, the size of a BFMap increases with number of its entries in a linear way, and thus they are not suited to calculate and exchange aggregates of several thousand entries.

8. CONCLUSION

We present our BSMX system that can be utilized to create novel applications without complex device configuration or significant impairment of the device's main functions. Our approach also allows the development of hybrid applications that communicate via WLAN and Internet. A sample application could be to exchange public IP addresses via WLAN for further communication or to detect the closeness of other devices without GPS and central server. Furthermore, we presented a novel probabilistic data structure that can be utilized to aggregate sensor data in a distributed manner. The data structure is also used by our prototype to maintain a multi-hop neighbor table which is accessible to subscribed applications. In future work, we will improve our prototype implementation and make it available under an open source license.

9. REFERENCES

[1] M. Caliskan, D. Graupner, and M. Mauve. Decentralized discovery of free parking places. In *VANET '06: Proceedings of the 3rd international workshop on Vehicular ad hoc networks*, pages 30–39, New York, NY, USA, 2006. ACM.

[2] J. Considine, F. Li, G. Kollios, and J. Byers. Approximate aggregation techniques for sensor databases. *Data Engineering, 2004. Proceedings. 20th International Conference on*, pages 449–460, April 2004.

[3] P. Flajolet and G. N. Martin. Probabilistic counting algorithms for data base applications. *J. Comput. Syst. Sci.*, 31(2):182–209, 1985.

[4] C. Lochert, B. Scheuermann, and M. Mauve. Probabilistic aggregation for data dissemination in vanets. In *VANET '07: Proceedings of the fourth ACM international workshop on Vehicular ad hoc networks*, pages 1–8, New York, NY, USA, 2007. ACM.

[5] T. Nadeem, S. Dashtinezhad, C. Liao, and L. Iftode. Trafficview: traffic data dissemination using car-to-car communication. *SIGMOBILE Mob. Comput. Commun. Rev.*, 8(3):6–19, 2004.

[6] B. Scheuermann and M. Mauve. Near-optimal compression of probabilistic counting sketches for networking applications. In *Dial M-POMC 2007: Proceedings of the 4th ACM SIGACT-SIGOPS International Workshop on Foundation of Mobile Computing*, Aug. 2007.

[7] S. Schnaufer, S. Kopf, H. Lemelson, and W. Effelsberg. Beacon-based short message exchange in an inner city environment. In *9th IFIP Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net 2010)*, June 2010.

[8] L. Wischhof, A. Ebner, H. Rohling, M. Lott, and R. Halfmann. Sotis - a self-organizing traffic information system. *Vehicular Technology Conference, 2003. VTC 2003-Spring. The 57th IEEE Semiannual*, pages 2442–2446 vol.4, April 2003.

[9] I. H. Witten, R. M. Neal, and J. G. Cleary. Arithmetic coding for data compression. *Commun. ACM*, 30(6):520–540, 1987.

EERS: Energy Efficient Responsive Sleeping on Mobile Phones

Bodhi Priyantha, Dimitrios Lymberopoulos, Jie Liu
Microsoft Research
One Microsoft Way, Redmond, WA 98052
{bodhip, dlymper, liuj}@microsoft.com

ABSTRACT

We present the concept of *Responsive Sleeping*, where the mobile phone can continue to sense its environment in the sleep mode. This capability enables novel applications in user interaction, context awareness, and people monitoring. However, the state of the art phone architectures today require the application processor (AP) to be active to read sensor data. The energy consumption makes responsive sleep impractical. In this paper, after analyzing the root cause of mobile sensing energy consumption, we present a new mobile phone architecture that uses heterogeneous multi-processors (or a multi-core processor) to achieve energy efficient responsive sleeping (EERS). We present a prototype called LittleRock and evaluate its energy benefit for responsive sleep through continuous sensing examples.

1. INTRODUCTION

Mobile phones today follow their users in almost every activity they engage in. The ubiquity, mobility, and connectivity of smart phones have made them ideal platforms for personalized mobile services, evident from the large number of applications available for various mobile platforms. Apart from having a reasonably powerful processor and graphics capability, current high-end smart phones have a rich set of built-in sensors, such as GPS, microphone, camera, accelerometer, ambient light, compass, gyro, and pressure sensors that enable measuring various phenomena on and around the phones and thus their owners. Location-aware services, natural user interfaces, and many games rely on these sensors to provide a superior user experience.

While it is most common for mobile applications to use the sensors *on-demand*, where sensor reading operations are initiated by the foreground process, application developers have also shown the value of performing continuous sensing in the background, even when the phone is not receiving direct user attention. For example, in *UbiFit* [3], Klasnja1 et. al. studies user behavior changes when they are fed with fitness information such as exercise tracked by wearable accelerometer and barometer sensors. In [6], a single wearable sensor is used to estimate the calori expenditure of a user. In *Playful Bottle* [2], Chiu et. al. attached a mobile phone to a water drinking device and used images and motion to detect the frequency and quantity of the user's water consumption. In SoundSense [8], Lu et. al. designed a mobile phone application that uses sound signatures to detect user location and activity. In the Mobile Millennium project [1], mobile phones were used as tripwire sensing devices to collectively detect traffic conditions.

All of these applications require the phone to continuously sample its sensors independently of whether the user is interacting with the phone. We call the capability that the phone can continuously sense while appears to be sleeping *Responsive Sleeping* (RS). In a responsive sleeping mode, the phone does not draw user attention or require a foreground application. However, in the background, it can continuously monitor sensor input and wake up the phone if interesting events happen.

Responsive sleeping can also greatly improve user experience with the phone, as if the phone is always on. For example, if a phone can detect that it is being picked up from the table and is approaching the face, it can automatically enter the voice command mode. If the phone can continuously monitor audio input, it can run speaker recognition to detect the user's company; then it can use this information to pop up reminders or set UI preference (e.g. adjusting ring tone or vibration). Another example is "geo-fencing", where an application registers a geographical region of interest, so that an event is triggered to wake up the phone and activate the application when the phone enters that region.

Responsive sleeping can be based either on sensors on the phones, or on external sensors that communicate wirelessly with the phone. In both cases, the energy consumption for supporting RS is challenging. For example, we will show in section 2 that a simple pedometer application can drain a smart phone battery in a few hours. This energy inefficiency is not due to the system software overhead, e.g., task scheduling, but rather due to the fundamental limitations of current phone architecture. That is, to acquire any sensor data or communicate with detached sensors, the application processor (AP) must be active and running. Active APs typically consume hundreds of milli-Watts (mW) of power. In section 2, we also show that sensor duty cycling [10] does not solve the energy inefficiency problem, since it takes up to a second for the AP to wake up from the sleep mode and restore the state for taking sensor readings.

Enabling *energy efficient responsive sleeping* (EERS) motivates us to rethink the mobile phone sensing architecture. In this paper, we propose a new mobile phone sensing architecture that can support EERS by breaking the tight coupling between sensors and the AP. With heterogeneous multi-processors or multi-core processors, a low power microcontroller or processor core can manage the sensors without shortening the battery life noticeably. As a prototype platform, we designed LittleRock, which adds a small, energy efficient co-processor to the phone to offload sensing tasks to this small processor. All the available sensors and

Sensor	Sensor State			HTC Touch Pro State (mW)		
	Active(mW)	Sleep(μW)	1Hz Sampling(mW)	Active (1680)	Idle (399)	Sleep (7.56)
Accelerometer	0.6	3	0.018	0.001%	0.004%	0.23%
Temperature	0.225	3	0.02	0.001%	0.005%	0.26%
Pressure	1.8	0.3	0.02	0.001%	0.005%	0.26%
Compass	2.7	7.5	0.5	0.03%	0.125%	6.61%
Gyro	19.5	15	1	0.06%	0.251%	13.22%
GPS chip (1MIPS CPU required)	214 (acq. state)	5	30	1.78%	7.518%	397%
Total (with GPS)	238.825	34.1	31.558	1.88%	7.91%	417%
Total (without GPS)	24.825	29.1	1.558	0.093%	0.39%	20.6%

Table 1: Power consumption of different types of sensors and their overhead (assuming 1Hz sampling rate) on the overall power consumption of an HTC Touch Pro phone in 3 representative power states.

short range radios on the phone are connected to the small processor, enabling the rest of the phone to go into the sleep mode. While the phone is sleeping, the co-processor can continue to acquire samples, process sensor data, and communicate with external sensors, all at a low energy overhead. Since the two processors are tightly integrated, data between them can be easily buffered and quickly exchanged on demand. We discuss EERS architectures in section 3 and the design of LittleRock in section 4.

Multiple sensing modalities can usually achieve the same goals with different energy and processing requirements. The addition of the sensing processor complicates this problem by introducing additional trade offs between energy and computation. In addition, the heterogeneous multiprocessor architecture brings complexities to software design and application programming. We discuss key hardware and software challenges toward responsive sleeping in section 6.

2. MOBILE ENERGY BREAKDOWN

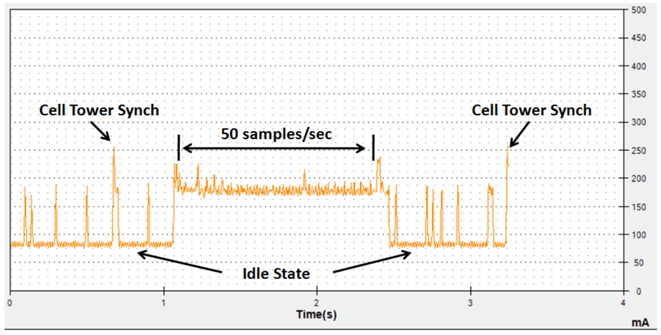
Battery life is one of the most critical design parameters for a phone. Every new feature introduced, either it is hardware or software, has to minimize its impact on the battery life. Consequently, although continuous sampling and processing of sensor data enables new application modalities, it is necessary that these additional features do not severely reduce the phone battery life.

Table 1 shows the overhead introduced by popular types of sensors in the power consumption of a mobile phone, the HTC Touch Pro running Windows Mobile 6.1. The power overhead for each sensor is expressed as a percentage of the power consumed by the HTC phone in 3 representative power states: *Active*(1680mW), *Idle*(399mW), and *Sleep*(7.56mW). In the *Active* state, the phone is exercising its CPU by running random computations while simultaneously downloading data over the 3G radio. In the *Idle* state the phone is turned on, but there is no load imposed on the CPU beyond the background services introduced by the operating system. Also, no data is being sent or received over the 3G radio. In the *Sleep* state the main processor is in sleep mode.

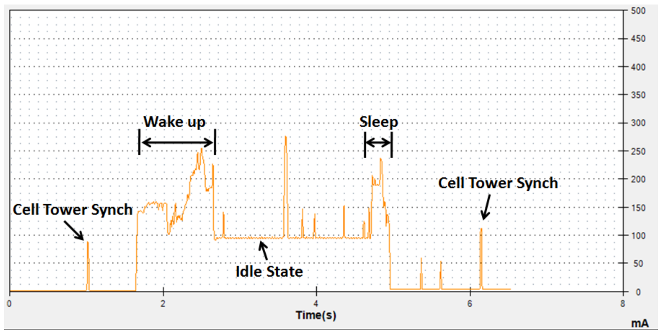
When all the sensors listed in Table 1 are powered up, the overall power consumption of the phone at the *Active*, *Idle* and *Sleep* states increases by approximately 1.88%, 7.91%, and 417% respectively. Note that in all cases, the GPS sensor is responsible for 95% of the overall power overhead. However, as recent work has demonstrated, more energy efficient location sensing can be achieved by properly combining cell tower triangulation and wifi fingerprinting to enforce more aggressive duty cycling of the GPS sensor [7],[9],[12]. Without the GPS sensor, the total sensor power overhead at the *Active*, *Idle* and *Sleep* states becomes 0.093%, 0.39%, and 20.6%.

Even though the continuous operation of the hardware sensors comes at a low power overhead, the process of accessing and processing sensor data on current state-of-the-art phones is extremely expensive. The reason is that for every sensor sample acquired by the phone, the main processor and associated components have to be active, creating a large energy overhead. To better illustrate the impact of continuous sensing on the battery life of current phones, consider an example application where the accelerometer on the phone is continuously sampled at a fixed frequency to perform a variety of tasks such as user activity recognition, dead reckoning-based indoor navigation, and step counting (pedometer) [5, 11, 4]. Figure 1(a) shows the power consumption of an HTC Touch pro phone while sampling the built-in accelerometer at the rate of 50 samples per second. When sampling the accelerometer, the overall power consumption of the phone jumps to approximately 756mW compared to the 7.56mW and 399mW of power consumption of the phone in the *Sleep* and *Idle* states respectively. This increase in power consumption is due to the fact that the CPU of the phone has to be active in order to acquire and store each accelerometer sample. In practice, this means that when sampling the sensors, the phone has to consume approximately 756mW, which is two orders of magnitude higher than the power consumed by the phone in the *Sleep* state.

Besides increasing the power consumption due to sampling, continuous sensing introduces another major bottleneck by essentially preventing the phone from moving to its *Sleep* state. The reason can be clearly seen in Figure 1(b), which shows the power trace for waking up and putting a phone into sleep. The phone needs approximately 900ms to move to and 270ms to exit from the *Sleep* state. As a result, a full transition between the phone's *Sleep* and *Idle* states takes more than a second. Because of this overhead, even when continuous sampling is required at a very low sampling rate, such as 2 samples per second, the phone does not have enough time to transition to and recover from the *Sleep* state and still acquire the next sensor sample on time. As a result, in order to meet the timing requirements for continu-



(a)



(b)

Figure 1: Current drawn from the HTC Touch Pro while (a) sampling the accelerometer at a rate of 50 samples per second and (b) performing a full sleep cycle.

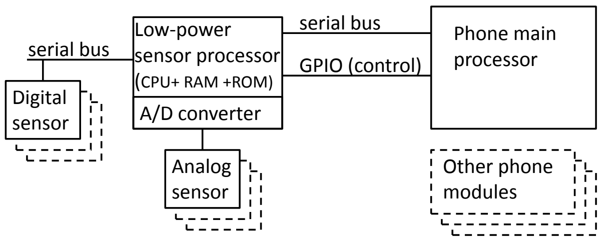


Figure 2: EERS phone architecture block diagram

ous sensing, the phone has to be *Active*, consuming approximately 756mW. Given the 1340mAh battery capacity, an HTC Touch Pro that continuously samples its accelerometer would last for ≃6.7 hours, ignoring other services such as phone calls, SMS send/receive, and 3G data traffic. With these other services, the battery life will reduce even further, making continuous sampling and processing of sensor data impractical on current phones.

The goal of EERS is to enable continuous sampling and processing of sensor data without significantly impacting phone's battery life.

3. SYSTEM ARCHITECTURE

To achieve EERS, continuous sampling and, to an extent, processing sensor data must be decoupled from the application processor of the phone. To achieve this, one can introduce a low power microcontroller, or a low power core in the multi-core processor to manage the sensors. This low power microcontroller or the processor core enables most parts of the phone to enter a low power *sleep* state, while the low-power sensor processor is continuously sampling and processing sensor data at a low-power overhead.

Figure 2 shows the block diagram of the proposed sensing architecture. In this architecture, various low-power analog and digital sensors are attached to a sensor processor. The sensor processor is typically a low-power microcontroller that consists of a CPU, RAM, ROM, and various peripherals such as serial communication buses. The sensor processor is interfaced to the main phone processor using a serial bus and multiple control signals.

3.1 EERS Architecture Features

We highlight the following benefits of introducing a low power sensing processor in EERS architecture:

Low power operation. Since a low-power processor with a power consumption similar to that of a typical sensor is used as the sensor processor, waiting during sensor readings and control does not impose high energy overhead. For example, the MSP430 family of low-power processors consumes $\simeq$ 1mW of power when operating at 1MHz.

Due to the simpler hardware architecture, a low-power processor can transition between sleep and active modes within a very short time. For example, the MSP430 class of processors can switch between the sleep ($\simeq$ 0.01mW) and active (16MHz clock, $\simeq$ 20mW) states in $<5\mu$s. This short transition time enables the sensor processor to be heavily duty cycled to reduce the average power.

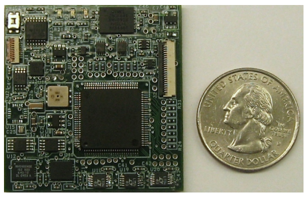
Programmable context-aware decision making. EERS provides a programmable approach to deciding when to wakeup the main processor based on sensor data. This decision making can be customized to meet the demands of the current user and the set of applications running on the phone. Since the sensor processor has access to nearly simultaneous readings from multiple sensors, the proposed architecture enables richer context aware decision making through sensor fusion.

Real time sensing and event handling. A typical microcontroller has multiple hardware modules such as counters, timers, A/D converters, and serial buses that can operate simultaneously. This hardware parallelism and the typical multi MHz processor clock speeds make it possible to achieve almost real time response when sampling and processing sensor data (assuming a light-weight processing workload).

3.2 Implementation Options

There are three options for introducing the proposed sensing architecture in mobile phones.

Use an existing microcontroller. Phones already have dedicated microcontrollers implementing specific functions. One good example is the capacitive touch controller which interprets various user touch events and communicates to the main processor using a serial bus. One possible implementation to enable EERS is to enhance one of these processors and attach the sensors to this processor. This can be a relatively low cost implementation option, since this requires only a simple modification to existing phone platform architecture.


(a)

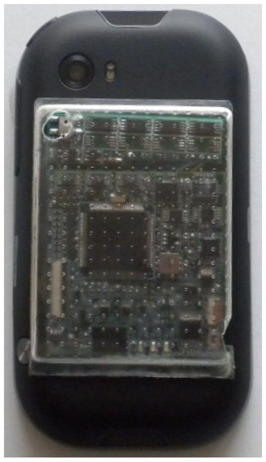

(b)

Figure 3: (a) The *LittleRock* prototype (b) The *LittleRock* prototype attached to the back of a phone.

Add a microcontroller. In this option, a dedicated low-power microcontroller is added to the phone. This option increases the total number of components on the phone. This is likely to be the most costly option out of the 3 mentioned here. However, given most low-power processors are relatively inexpensive (<$1) in typical cellphone volumes, the new application modalities enabled by EERS are likely to offset this additional cost.

Add a low-power core. This option adds a low-power core to the phone's main processor. This is likely to be the least cost option, since incremental cost of adding a simple, low-power core on to the phone's main processor is likely to be very small. However, this option is not likely to be the first option adopted by phone manufacturers, since this requires changes to the phone's main processor itself.

4. PROTOTYPE

We prototyped and evaluated the EERS sensing architecture using LittleRock, a prototype hardware platform consisting of multiple sensors and a low-power processor that interfaces directly to the main processor of a prototype phone. The main goals of this prototype were to experiment with various context-aware applications that can benefit from EERS, and to evaluate the energy overhead due to the proposed sensing architecture.

Our prototype connects to the phone using a wired interface. The resulting platform has a form factor similar to that of a typical mobile phone, enabling us to conduct various user studies on various EERS-enabled applications. This section describes the details of LittleRock prototype.

LittleRock consists of four functional modules: the processor, digital sensor, analog sensor, and the phone interface.

The processor module consists of an MSP430F5438 processor with 16kB RAM and 256kB flash memory. The processor can be clocked up to 18MHz. This processor also has a large number of parallel and serial inputs and outputs, making it possible to attach additional sensors than those already built in to our prototype. This prototype also has a smaller MSP430 processor (MSP430F2013) for on demand reprogramming of the MSP430F5438 processor. The processor module also contains an 8MB flash storage.

Batch size	Power		
	Phone	LittleRock	Hybrid
10	693	0.21	702
50	693	0.21	171
100	693	0.21	90
500	693	0.20	25

Table 2: Average power required to acquire and process 500 samples at a 10Hz sampling rate at different processing batch sizes

The digital sensor module contains a temperature sensor, a 3-axis accelerometer, a barometer, and a 3-axis compass module connected to the main processor through an I2C bus.

The analog sensor module consists of sensors that have analog outputs. In particular, this module contains an X-Y axis gyroscope and a Z axis gyroscope that collectively provide 3-axis gyroscopic data. To reduce the impact due to processor generated digital noise, and to provide better resolution than what is possible with the built in A/D converter of the processor, we used 3 external 16 bit A/D converters to digitize the gyroscope outputs.

The phone interface consists of a wired connection between LittleRock and the expansion connector on a prototype phone. LittleRock and the phone communicate over a 4 wire SPI bus. LittleRock is directly powered from the phone's battery, through the expansion connector. The phone's IO voltage exposed by this connector indicates if the phone is powered on or off. A GPIO pin enables LittleRock to interrupt and wake up the main processor.

5. EVALUATION

In this section we evaluate the performance of a pedometer application that counts user steps based on periodic accelerometer samples. A pedometer is a classic application that benefits from EERS, where the accelerometer has to be continuously sampled even when the phone is in *sleep* mode.

The pedometer application samples a 3-axis accelerometer at 10 Hz, after collecting a batch of n samples, it examines the magnitude variation of acceleration to detect user step events. We evaluate the energy consumption of a pedometer application under three different configurations: running on the phone, running on LittleRock, and running on a hybrid of phone and LittleRock. In the hybrid approach LittleRock buffers the batch of n samples and sends them to the phone for updating the step count.

Table 2 shows the average power consumption of the three different hardware configurations. The column under "Phone" is the most energy inefficient configuration since it consumes $\simeq$ 700mW. Note that this number does not change with the processing batch size, since at the 10Hz sampling rate, the phone continues to be in the *active* state due to the large sleep transition time.

The LittleRock only configuration consumes $\simeq$ 0.2mW. This corresponds to more than 3 orders of magnitude improvement in the power consumption compared to the phone only configuration. Unlike the phone, the low-power processor on LittleRock can transition to sleep mode almost instantly. This, combined with the lower power consumption of the processor, reduces the overall power overhead in LittleRock configuration. Note that, in this configuration, the power consumption reduces slightly with larger batch sizes due to the amortization of fixed processing overhead over a larger batch of data.

Table 2 also shows that the hybrid approach can be significantly energy efficient for large batch sizes. This is because larger batch sizes enables the phone to spend more time in the sleep mode, while LittleRock is sampling and buffering sensor data.

6. RESEARCH CHALLENGES

The proposed architecture for enabling EERS fundamentally changes how applications interact with sensors. This brings additional systems challenges.

Sensor processor selection. In the proposed architecture, sampling and processing of sensor data is offloaded to a sensor processor or a sensor processor core. However, determining how much processing capability and functionality the sensor processor should possess is a challenge.

If the sensor processor does not have enough processing capability, the main processor has to be woken up more often, resulting in high energy consumption. On the other hand, too much functionality in the sensor processor increases the processor sleep currents and wakeup times, and thus the average power. Consequently, the processor selection requires a careful evaluation of the anticipated resource requirements.

Sensor processor resource management. The sensor processor samples and processes sensor data on behalf of multiple user applications. This can result in multiple user applications competing for resources on the sensor processor. Such competing applications raise two challenges.

The first challenge is ensuring fair sharing of sensor processor resources among multiple applications. The second challenge is preventing applications from overloading the sensor processor, which can lead to unpredictable behaviors due to effects such as stack overflows. To address these concerns, the sensor processor needs to employ strict resource counting and management of its memory, processing, and energy resources.

Sensor API. The user applications access the sensor sampling and processing services of the sensor processor through a sensor API. This API should be flexible enough to access multiple services provided by a generic programmable processor, while being simple enough for application developers to use this API without spending too much effort.

Application partitioning. Under the proposed architecture, an application that uses sensor data spans across the sensor processor and the main processor. With this, an application developer needs to decide how to partition an application across these processors. Offloading too little to the sensor processor results in spending too much energy on the main processor, while offloading too much can overburden the sensor processor. Enabling application developers to make the best decision on how to partition an application will require support for fine-grained application profiling.

7. CONCLUSION

This paper focuses on *Responsive Sleeping*, where data from multiple sensors attached to the phone are continuously sampled and processed, even when the phone appears to be in the sleep mode. RS is an advanced feature that enables novel user interface, participatory sensing, and health and fitness applications.

Through detailed measurements we show that the current phone architecture, where all sensors are directly controlled by the phone processor, cannot meet the battery lifetime requirements for RS.

Based on these results, we propose a new phone sensing architecture for energy efficient responsive sleeping (EERS) on phones, where the sampling and processing of sensor data is offloaded to a low-power sensor processor. Using a step counting application running on the LittleRock prototype, we show that the proposed sensing architecture enables energy efficient *Responsive Sleeping* on phones.

8. REFERENCES

[1] B.Hoh, M. Gruteser, R. Herring, J. Ban, D. Work, J.-C. Herrera, A. Bayen, M. Annavaram, and Q. Jacobson. Virtual trip lines for distributed privacy preserving traffic monitoring. In *Mobisys'08*, June 2008.

[2] M.-C. Chiu, S.-P. Chang, Y.-C. Chang, H.-H. Chu, C. C.-H. Chen, F.-H. Hsiao, and J.-C. Ko. Playful bottle: a mobile social persuasion system to motivate healthy water intake. In *Ubicomp '09*, 2009.

[3] S. Consolvo, P. Klasnja, D. W. McDonald, D. Avrahami, J. Froehlich, L. LeGrand, R. Libby, K. Mosher, and J. A. Landay. Flowers or a robot army?: encouraging awareness & activity with personal, mobile displays. In *UbiComp '08*, 2008.

[4] R. C. Foster, L. M. Lanningham-Foster, C. Manohar, S. K. McCrady, L. J. Nysse, K. R. Kaufman, D. J. Padgett, and J. A. Levine. Precision and accuracy of an ankle-worn accelerometer-based pedometer in step counting and energy expenditure. *Preventive Medicine*, 41(3-4):778 – 783, 2005.

[5] J. Lester, T. Choudhury, and G. Borriello. A practical approach to recognizing physical activities. In *In Proc. of Pervasive*, pages 1–16, 2006.

[6] J. Lester, C. Hartung, L. Pina, R. Libby, G. Borriello, and G. Duncan. Validated caloric expenditure estimation using a single body-worn sensor. In *Ubicomp '09*, 2009.

[7] K. Lin, A. Kansal, D. Lymberopoulos, and F. Zhao. Energy-accuracy trade-off for continuous mobile device location. In *MobiSys'10*, June 2010.

[8] H. Lu, W. Pan, N. D. Lane, T. Choudhury, and A. T. Campbell. Soundsense: Scalable sound sensing for people-centric sensing applications on mobile phones. In *Mobisys'09*, June 2009.

[9] J. Paek, J. Kim, and R. Govindan. Energy-efficient rate-adaptive gps-based positioning for smartphones. In *MobiSys'10*, June 2010.

[10] Y. Wang, M. A. Jialiu Li and, Q. Jacobson, J. I. Hong, B. Krishnamachari, and N. Sadeh. A framework of energy efficient mobile sensing for automatic user state recognition. In *MobiSys'09*, June 2009.

[11] O. Woodman and R. Harle. Pedestrian localisation for indoor environments. In *UbiComp '08*, 2008.

[12] Z. Zhuang, K.-H. Kim, and J. P. Singh. Improving energy efficiency of location sensing on smartphones. In *MobiSys'10*, June 2010.

Pocket, Bag, Hand, etc. - Automatically Detecting Phone Context through Discovery

Emiliano Miluzzo[†], Michela Papandrea[§], Nicholas D. Lane[†], Hong Lu[†], Andrew T. Campbell[†]

[†]CS Department, Dartmouth College, Hanover, NH, USA

[§]SUPSI, Manno, Switzerland

ABSTRACT

Most top end smart phones come with a handful of sensors today. We see this growth continuing over the next decade with an explosion of new distributed sensor applications supporting both personal sensing with local use (e.g., healthcare) to distributed sensing with large scale community (e.g., air quality, stress levels and well being), population and global use. One fundamental building block for distributed sensing systems on mobile phones is the automatic detection of accurate, robust and low-cost *phone sensing context*; that is, the position of the phone carried by a person (e.g., in the pocket, in the hand, inside a backpack, on the hip, arm mounted, etc.) in relation to the event being sensed. Mobile phones carried by people may have many different sensing contexts that limit the use of a sensor, for example: an air-quality sensor offers poor sensing quality buried in a person's backpack. We present the preliminary design, implementation, and evaluation of *Discovery*, a framework to automatically detect the phone sensing context in a robust, accurate and low-cost manner, as people move about in their everyday lives. The initial system implements a set of sophisticated inference models that include Gaussian Mixture Model and Support Vector Machine on the Nokia N95 and Apple iPhone with focus on a limited set of sensors and contexts. Initial results indicate this is a promising approach to provide phone sensing context on mobile phones.

1. INTRODUCTION

The recent explosion of smartphones (e.g., Nokia, Apple iPhone, and Android phones) with embedded sensors is enabling a new generation of personal and environmental sensing applications [1, 2, 3, 4]. These applications are built on multi-faceted real-time sensing operations that require increasing computation either on the phone [2] or backend servers [3], or a combination of both [1]. As the demands of these new distributed sensing applications built on commercial phones is better understood in terms of their needs for on-phone sensors, computation and communication resources, a number of important challenges are emerging. Because these continuous sensing applications are extremely resource hungry in terms of sensing, computation and communications (with backend servers) there is need to drive the operation of the phone in a more intelligent manner. We believe efficiently computing the low level context of the phone, that is, the position of the phone carried by a person (e.g., in the pocket, in the hand, inside a backpack, on the hip, arm mounted, etc.) in relation to the event being sensed - which we call the *phone sensing context* - is a fundamental building block for new distributed sensing applications built on mobile phones. These observations have grown out of our implementation of CenceMe [1] and SoundSense [2], two continuous sensing applications implemented on Nokia and Apple phones. While there has been significant research in the area of context aware applications and systems, there has been little work on developing reliable, robust, and low cost (i.e., in terms of energy efficient and computational costs) algorithms that automatically detect the phone sensing context on mobile phones. We envision a future where there are not only personal sensing applications but we see the mobile phone as enabling global sensing applications where the context of the phone in relation to the sensing event is crucially important.

The different context impacts the fidelity of a sensing application running on mobile phones. For example, the camera is of little use in the pocket but the microphone might still be good [2]. Researchers are developing new sensors for the phones that we imagine will be available over the next decade, these include CO_2 and pollution sensors [5]. If the phone is carried inside the pocket or a backpack, an application relying on CO_2 or pollutants measurements would perform very poorly given that the phone is not exposed to open air. A better position for such sensing would be out of the pocket when the phone is exposed to a more suitable context for sensing. Similarly, if the accelerometer readings of the phone are used to infer the person's activity, the accelerometer would report different data if the phone is mounted on the arm or clipped to the belt. This is because, given the same activity, such as walking for example, arm swings would activate the accelerometer much more strongly for an arm-mounted phone than on the belt, where the phone oscillates more gently. In both cases a mechanism to infer the context of the mobile phone is needed in order to make the applications using the CO_2 or pollution sensor and the accelerometer, respectively, react appropriately. We envision a learning framework on the phone that is more sophisticated than what is implemented today. For example, when sensors report different sensor readings according to the position on the body, such as the accelerometer, the application's learning engine should switch to different classification algorithms or sensor data treatment policy in order to meet the application requirements.

Today the application sensing duty-cycle is costly because it is not driven by the phone sensing context, therefore, it is costly in terms of energy usage for sensing, computation and potentially communications if the inference is done on the backend, as in the case with split-level classification [1]. By offering system developers accurate phone sensing context prior to running classification algorithms, very low duty-cycle continuous sensing application systems are possible. In this case, the phone sensing context mechanism would refrain the application from activating a power hungry sensor if the context is unsuitable (e.g., don't activate the pollution sensor if the phone is not out of the pocket) or it may weight real-time sensor readings or inferences based on knowledge of where the phone is on the body (e.g., if the microphone is needed to measure human activity [2] and it is in the bag).

In this paper, we discuss Discovery, a framework that addresses the context problem supporting mobile phone-based sensing with improved accuracy and lower duty-cycle systems. Discovery is designed to automatically detect the phone sensing context as people move about in their everyday lives. Automatic context detection is a primary issue for mobile phone sensing applications because prompting the user to provide information about the position of the mobile phone on the body is not a viable and scalable solution. Phone sensing context is an important building block toward the successful implementation of personal, social, and public sensing applications on mobile phones and the work in this paper, while preliminary, provides important steps towards the goal of providing reliable phone sensing context. This paper is organized as follows. Section 2.1 contains the motivation of this work, while details of the approach taken in Discovery are discussed in Section 2.2. Preliminary evaluation results are discussed in Section 3. Future directions are reported in Section 4 and the related literature in Section 5, before concluding in Section 6.

2. DISCOVERY FRAMEWORK

In what follows, we discuss some challenges phone sensing context presents, its preliminary design and implementation as part of the Discovery framework, as shown in Figure 1.

2.1 Phone Sensing Context

Accurate, robust and low duty-cycle detection of phone sensing context is an important enabler of distributed sensing applications on phones, in particular, continuous sensing applications that sample sensors, make inferences, and communicate with the backend services in real-time.
Assume mobile phones are equipped with pollution, CO_2, or more specialized environmental sensors as we imagine [5]. Measurements from any of these sensors would most likely be impeded by the presence of clothing or fabric (e.g., phone inside the pocket or backpack) or by a short time interval the sensors are exposed to an ideal sensing context (i.e., phone in hand or exposed to open air). Therefore, phone sensing context detection would improve the sensing system performance. We could stop the system from activating the sensors when the quality of the sensor data is likely to be poor (e.g., phone inside the pocket). This would help reduce the sensing duty-cycle improving the battery lifetime of the phone, which continuos sensing application significantly limit today (e.g., phones running CenceMe [1] were initially limited to only 6 hours of operation). We could inform the system when a suitable sensing context is triggered or detected (e.g., phone taken out of the pocket) to maximize the accuracy and robustness of the sensing application which would then take advantage of the new context for collecting as many sensor readings as possible. It is evident the importance of the phone sensing context role in driving mobile phones sensors duty-cycle lower.

Another reason to provide phone sensing context as a low level service on phones is to improve the inference fidelity of distributed sensing applications. Although previous work [6] shows that it is possible to obtain reasonably good activity classification accuracy when using training data from sensors mounted on different parts of the body, it is not clear how an activity classifier would perform when the device is a phone, not specifically mounted (but moving as a dynamic system), and operates in noisy, everyday environments that people find themselves in, rather, than under laboratory test conditions. Many questions remain. Would training data from many activities and different parts of the body make a single classification model accurate enough? To avoid excessively diluting the training data set, would it not be preferable building a classification model for each single activity and position of the mobile phone on the body and then switch models according to the detected phone sensing context? For example, a system could have a "walking" activity classification model for when the mobile phone is in the pocket, in the person's hand, and in the backpack and use one of the models according to the detected phone sensing context. Results obtained from experimentation in [1] show, for example, that activity classification accuracy varies when the phone is carried in the pocket or in the hand. A system that used phone sensing context to drive the classification model by switching in the right technique would alleviate this problem. We believe this is of importance now that smart phones are growing in sensing and computational capability and new demands are emerging from different sectors such as healthcare. It is important to note that in the case of health care sensing applications it is fundamental to limit the classification error. Sensing context detection could drive inference model switching in order to achieve better classification accuracy.
We argue that phone sensing context detection could also be exploited by existing phone applications and services. For example, by inferring that the phone is in the pocket or bag, a caller might be informed about the reason the callee is not answering the phone call while the callee's phone ring tone volume could be increased so the callee might pick up. One could imagine people enabling this type of additional presence provided to legacy phone service through Discovery. By using the gyroscope (which measures the angular rate change of the phone) to detect the user taking the phone out of the pocket and moving it upwards, the screen saver could be disabled and the phone's keypad made automatically available. One could imagine many such adaptations of the UI with phone sensing context enabled. Similarly, the action of moving the phone towards the lower part of the body could trigger power saving mode. The camera application on the phone could be automatically started as the phone is detected in the user's hand and moved in a vertical position, which is the condition that normally precedes the action of taking a photo. One could imagine phone sensing context provided by the Discovery framework discussed in the next section being applicable to many emerging applications finding their way on to smartphones. For example, reality mining using mobile phone sensor data is starting to be explored as an enhanced form of communication and for social purposes [7].

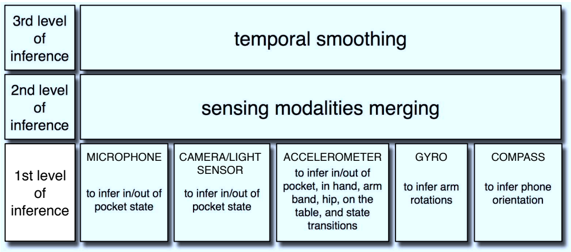

3rd level of inference
temporal smoothing
2nd level of inference
sensing modalities merging
1st level of inference
MICROPHONE
to infer in/out of pocket state
CAMERA/LIGHT SENSOR
to infer in/out of pocket state
ACCELEROMETER
to infer in/out of pocket, in hand, arm band, hip, on the table, and state transitions
GYRO
to infer arm rotations
COMPASS
to infer phone orientation

Figure 1: Discovery inference steps.

2.2 Design

The idea behind Discovery is to use the entire suite of sensing modalities available on a mobile phone to provide enough data and features for context discovery at low cost and for increased accuracy and robustness. Many research questions arise in response to the challenges discussed above: how do we combine the input from multiple sensors, such as, accelerometer, microphone, gyroscope, camera, compass, etc., to infer the phone sensing context? What are the best learning approaches and feature selection policies in order to provide a reliable and scalable context inference system? How do we design low duty-cycling policies with acceptable accuracy when employing phone sensing context? What is the inference accuracy and energy cost tradeoff between using all the possible sensors and only a subset of them according to their availability on the mobile phone? Which sensor set is more responsive to the type of noise in the system (i.e., classification outside controlled laboratory environments)? We believe that Discovery in its totality needs to ultimately address these demanding challenges. However, our preliminary work focuses on a simple phone sensing context: is the phone in the pocket or out. This sounds like a trivial context that could be solved by a number of different sensors. We focus on the microphone - a powerful and ubiquitous sensor on every phone on the market - making Discovery suitable to potentially all phones not just the smart ones. In what follows, we outline out initial framework design.

Discovery consists of a hierarchical inferences pipeline, as illustrated in Figure 1:

First Level Inference - Uni-sensor inference: In this phase, the sensor data from individual sensors is used to operate a first level of inference. Features extraction is tailored to each sensor. This first inference step provides hints about the nature of the current phone sensing context, which, however, might not be conclusive. For example, the use of the camera or light sensor to infer if the phone is in or out the pocket could be misleading because a phone out of the pocket could be in a dark environment, the camera could be covered by the person's hand or by the surface where the phone is positioned. For this reason, a second level of inference built on top of the first is needed.

Second Level Inference - Multi-sensor inference: In this phase, the inference process is based on the output of the first phase. Hence, the first level of inference provides the features to the second level. At this stage, the combination of the camera/light sensor and microphone output would provide better confidence about the actual sensing context. The accelerometer as well could be used as a hint to determine if the phone is inside or outside the pocket given the different accelerometer data signatures when the phone is in a person's hand versus when it's in the pocket. Similarly, by measuring the angular rate change, the gyro could provide indications that the phone has been taken out of the pocket considering that the arm rotation would be picked up by the gyroscope.

Third Level Inference - Temporal smoothing: In this phase, temporal smoothing and Hidden Markov Model (HMM) techniques are used on the output of the second level inference. This step exploits the correlation in time of sensed events when a phone experiences a certain context.

2.3 System Implementation

For our initial implementation of Discovery context classifiers are implemented on the Nokia 95 and Apple iPhone. The preliminary system implements a set of sophisticated inference models that include Gaussian Mixture Model (GMM) and Support Vector Machine (SVM) on the Nokia N95 and Apple iPhone with focus on a limited set of sensors and inferences; that is, we uses the microphone sensing modality to infer the phone sensing context of in the pocket and out of the pocket. We discuss our initial results in the next section. Further modalities, such as accelerometer, compass, and light sensor, are going to be used in combination with the microphone to infer a larger set of sensing context as part of our future work. The initial idea is to evaluate which learning technique (between GMM and SVM) is better suited to the problem and, at the same time, to investigate the adoption of more than one learning strategy in concert to perform the final classification. More learning strategies will be evaluated in the following phase of this work. The challenge with GMM and SVM is that the phone has not been developed to run these computationally demanding models. Part of our efforts is to implement light weight versions of these models as a way forward to do more sophisticated multi-inference classification, as called for by Discovery. In particular a 20-component GMM is adopted, where the number of components is chosen by evaluating the model over the test data set varying the number of components and picking the number of components returning the best classification accuracy.

Feature Selection. The selection of an appropriate set of features is a key step to good classification performance. At the moment, a supervised learning approach is adopted and Discovery relies on a 23-dimensional feature vector extracted from an audio clip. A richer selection of features will be evaluated as part of our future work. The current features are:

1st-19th: Mel-Frequency Cepstral Coefficients (MFCC), which have been proven to be reliable features in audio signal classification problems. For the MFCCs extraction we rely on a well-known Matlab libray [8] which is largely used by the research community. We also developed a C version of the MFFC extractor library that can run on the phone;

20th: power of the audio signal calculated over the raw audio data;

21st, 22nd: mean and standard deviation of the 2048-point FFT power in the 0-600 Hz portion of the spectrum. The reason for focusing on this portion of the spectrum can be seen from Figures 2(a) and 2(b), where the presence of a pattern between the two FFT distributions - for in pocket and out-of-pocket recording - is clear. It can be seen that such a pattern is more evident in the 0-600 Hz portion of the spectrum rather than in the whole 0-1024 Hz range;

23rd: this feature is the count of the number of times the

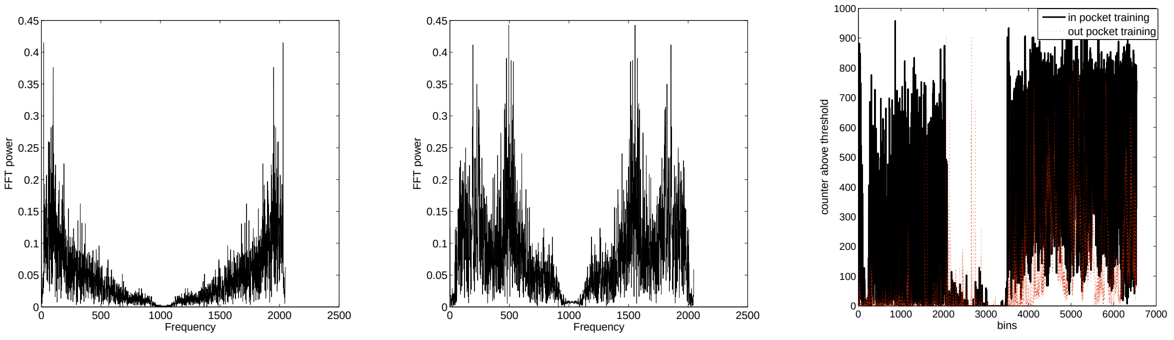


Figure 2: (a) FFT power of an audio clip when the phone is inside the pocket; (b) FFT power of an audio clip when the phone is outside the pocket; (c) Count of the number of times the FFT power exceeds a threshold *T* for both the in-pocket and out-of-pocket cases.

Table 1: Sensing context classification results using only the microphone. Explanation: when a result is reported in X/Y form, X refers to the *in pocket* case, and Y refers to the *out of pocket* case. If the column reports only one value, it refers to the average result for both *in* and *out* of pocket. Legend: A = GMM; B = SVM; C = GMM training indoor and evaluating indoor only; D = GMM training outdoor and evaluating outdoor only; E = SVM training indoor and evaluating indoor only; F = SVM training outdoor and evaluating indoor only; G = GMM training using only MFCC; H = SVM training using only MFCC.

Classification results	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**
Accuracy	84% / 78%	80%	75% / 84%	84% / 83%	68%	81%	77% / 79%	71%
Error	16% / 22%	20%	25% / 16%	16% / 17%	32%	19%	23% / 21%	29%

FFT power exceeds a certain threshold *T*. This threshold is determined by measuring the Euclidean difference between the count of the in-pocket and out-of-pocket cases and picking the threshold that maximizes such a distance. An example of the count for both the in-pocket and out-of-pocket cases is shown in Figure 2(c) where it can be seen how these features can be used to discriminate between the in pocket and out of pocket cases. The x-axis of Figure 2(c) reports the number of bins the clip has been split in to.

Consequently, for the mixture model, a 20-component, 23-dimensional GMM is used. The SVM classifiers adopts the 23 dimensional feature vector.

Training. The training phase is performed using audio data collected with a Nokia N95 and Apple iPhone in different settings and conditions from a person going through different environments for several hours. Namely, the audio is recorded in a quiet indoor office environment and an outdoor noisy setting (along a road with cars passing by). In both scenarios the phone is carried both in the pants pocket and outside the pocket in the hand. The choice of these scenarios, i.e., indoor and along a road, is motivated by the fact that they are representative of classes of locations where most likely people spend a lot of their time while carrying their phone both inside and outside the pocket. For each configuration 14 minutes of audio are recorded at different times. Half of each clip (i.e., about 7 minutes of audio) is used to train the classifiers. The training data is finally labeled accordingly.

Prediction. For prediction, the remaining half of each audio clip not part of the training set (i.e., duration of about 7 minutes) is used. Each sample consists of a 96 msec chunk from which the 23 features are extracted. For each configuration there are about 58000 samples available for training and 58000 for evaluation.

3. PRELIMINARY SYSTEM EVALUATION

In what follows, preliminary results from using both the GMM and SVM classification techniques are reported. The results highlight that the audio modality is effective in detecting the in/out of pocket context with reasonable accuracy. Higher accuracy can be achieved by combining further modalities such as accelerometer and light sensor. Columns A and B in Table 1 show, respectively, the classification results for GMM and SVM when the training data combines both indoor and outdoor audio and the phone is carried in and out the pocket. The results are quite encouraging, since we obtain about 80% accuracy (see the accuracy values in columns A and B) adopting a non sophisticated feature set and using only one sensing modality, i.e., the microphone. We are confident that by bringing into the classification process more modalities, for example the accelerometer and light sensor, a more accurate selection of the feature vector, and temporal smoothing it might be possible to achieve a much higher classification accuracy. We then train and evaluate the models for only one scenario, i.e., either indoor or outdoor. The results using GMM are in Table 1 column C and column D. The results for SVM are in column E and column F. In the case of SVM trained and evaluated for the indoor scenario only (see column E) the accuracy is lower than the other cases because Libsvm (the well known SVM library implementation we adopt) is running with the default settings with the kernel optimization being disabled. From these results it is interesting to see that training the models with both indoor and outdoor data does not dilute the training data and the final classification accuracy does not drop significantly compared to the case when the models are trained for a single scenario only and evaluated for the same scenario. In fact, the accuracy in columns C, D, and F is on average close to 80% as in the case of indoor and outdoor training data set (see columns A and B). Columns G and H in Table 1 show, respectively, the classification results for GMM and SVM when the model is trained using only MFCCs (hence a 19-dimensional feature vector). It is evident that the addition of the 4 extra features (i.e., signal power, FFT mean, FFT stddev, and number of times a threshold is exceeded by the FFT power) boosts the classification accuracy. The improvement can be seen by comparing the results in columns G and H with the ones in columns A and B.

4. FUTURE WORK

After the initial promising results, the goal is to implement a working prototype for the Android platform as well. More sensing modalities are going to be used in combination with the audio modality. In particular, the accelerometer, magnetometer, and light sensors. Research is going to be needed in order to identify the most suitable feature vector elements that combine the characteristics of all the sensing modalities. Temporal correlation between events is also going to be taken into consideration to improve the overall accuracy. Techniques such as HMM or voting strategies will be taken into account. We will also pursue the idea of letting people customize the Discovery classifiers to accommodate their habits and needs.

5. RELATED WORK

In the literature, context awareness follows the definition that Weiser [9][10] and others [11][12] provided when introducing or evolving ideas and principles about ubiquitous computing. In that case, context awareness is intended as either the awareness of situations and conditions characterizing sensor devices surroundings or the behavior, activity, and status of the person carrying the sensors in order to provide smart ways to facilitate and explore interaction between machines and humans. Thus, context is seen as the collection of happenings around a monitored subject and the response of the subject to such those happenings. The work in [13, 14, 15, 16, 14] are examples of how sensing systems are adopted to infer such a context and/or leverage context awareness. In some cases external sensors, i.e., not part of the mobile phone itself, are also needed [14][13] in order to perform accurate context inference. The authors of [17] use the word context to mean location awareness and propose applications that efficiently build on top of it. A very large body of work focuses instead on the use of various sensing modalities such as accelerometer, magnetometer, gyroscope to infer a person's activities for different applications [18, 19, 6, 20, 1, 21, 22, 23]. The authors in [24] present an approach to help discover the position of the phone on a person's body. The work highlights two limitations: it uses simple heuristics derived from a small training data set to determine the classification rules, and it uses a single modality approach, i.e., the accelerometer. We instead rely on a systematic design using machine learning algorithms that are more scalable and robust than simple heuristics and consider a larger training data set from multiple positions on the body and different scenarios while using a multi-sensing modality approach.

6. CONCLUSION

In this paper, we argued that phone sensing context is a key system component for future distributed sensing applications on mobile phones. It should be designed to be accurate, robust, and low cost. We discussed our initial work on the Discovery framework that grew out of our work on the deployment of two continuous sensing applications implemented and deployed on Nokia and Apple phones. Our initial implementation and evaluation only focuses on a limited set of sensors/contexts, but looks promising and, as an idea, it has potential, when implemented in its full form, to become a core component of future mobile sensing systems.

7. ACKNOWLEDGMENTS

This work is supported in part by Nokia, Intel Corp., Microsoft Research, and NSF NCS-0631289.

8. REFERENCES

[1] Miluzzo E. et al. Sensing Meets Mobile Social Networks: the Design, Implementation and Evaluation of the CenceMe Application. In *SenSys'08*, 2008.
[2] Lu H. et al. SoundSense: Scalable Sound Sensing for People-Centric Applications on Mobile Phones. In *MobiSys'09*, 2009.
[3] M. Mun et al. PEIR, the Personal Environmental Impact Report, as a Platform for Participatory Sensing Systems Research. In *MobiSys'09*, 2009.
[4] Azizyan M. et al. SurroundSense: Mobile Phone Localization via Ambience Fingerprinting. In *MobiCom'09*, 2009.
[5] Honicky R.J et al. N-SMARTS: Networked Suite of Mobile Atmospheric Real-Time Sensors. In *NSDR'08*, 2008.
[6] J. Lester, T. Choudhury, and G. Borriello. A Practical Approach to Recognizing Physical Activities. In *Pervasive'06*.
[7] Your mobile phone as an intelligent sensing machine. http://tinyurl.com/m3hgby.
[8] Rastamat. http://labrosa.ee.columbia.edu/matlab/rastamat.
[9] M. Weiser. The Computer for the 21 st Century. In *Mobile Computing and Communications Review*, 1999.
[10] M. Weiser. Some computer science issues in ubiquitous computing. In *Mobile Computing and Communications Review*, 1999.
[11] B. Schilit, N. Adams, and R. Want. Context-Aware Computing Applications. In *WMCSA'94*, 2004.
[12] P. Dourish. What we talk about when we talk about context. In *Personal and ubiquitous computing*, 2004.
[13] Siewiorek D. et al. Sensay: A context-aware mobile phone. In *ISWC'03*, 2003.
[14] Gellersen H.W. et al. Multi-sensor context-awareness in mobile devices and smart artifacts. In *Mobile Networks and Applications*, 2002.
[15] The Context Aware Cell Phone Project. http://www.media.mit.edu/wearables/mithril/phone.html.
[16] J.E. Bardram and N. Nørskov. A context-aware patient safety system for the operating room. In *Ubicomp'08*, 2008.
[17] Lukkari J. et al. SmartRestaurant: mobile payments in context-aware environment. In *ICEC'04*, 2004.
[18] Harrison B. et al. Using Multi-modal Sensing for Human Activity Modeling in the Real World. In *Handbook of Ambient Intelligence and Smart Environments*, 2010.
[19] Choudhury T. et al. The Mobile Sensing Platform: An Embedded Activity Recognition System. *IEEE Pervasive Computing*, pages 32–41, 2008.
[20] L. Bao and S.S. Intille. Activity recognition from user-annotated acceleration data. In *Pervasive'04*, 2004.
[21] Parkka J. et al. Estimating intensity of physical activity: a comparison of wearable accelerometer and gyro sensors and 3 sensor locations. In *EMBS 2007*.
[22] Li Q. et al. Accurate, Fast Fall Detection Using Gyroscopes and Accelerometer-Derived Posture Information. In *BSN'09*, 2009.
[23] S.W. Lee and K. Mase. Activity and location recognition using wearable sensors. In *IEEE Pervasive Computing*, 2002.
[24] Kawahara Y. et al. Recognizing User Context Using Mobile Handset with Acceleration Sensors. In *Portable'07*, 2007.

A Comfort Measuring System for Public Transportation Systems Using Participatory Phone Sensing [*]

Cheng-Yu Lin[1,2], Ling-Jyh Chen[1,2], Ying-Yu Chen[2], and Wang-Chien Lee[3]
[1] Research Center for Information Technology Innovation, Academia Sinica
[2] Institute of Information Science, Academia Sinica
{ntuaha, cclljj, ciy}@iis.sinica.edu.tw
[3] Department of Computer Science and Engineering, The Pennsylvania State University
wlee@cse.psu.edu

ABSTRACT

Participatory phone sensing is a new sensing paradigm that asks volunteers to contribute their phones' sensing capabilities and gather, analyze, and share local knowledge about their surroundings. While most existing participatory phone sensing systems are standalone structures without cross-system integration, in this study, we propose a novel Comfort Measuring System (CMS) for public transportation systems. CMS exploits the GPS and 3-axis accelerometer functions of modern smart phones to measure the comfort level of vehicle rides. Then, it mashes up the sensed data with the authorized data of the public transportation system, and provides a detailed comfort statistics as a value added service. Using real data collected from a CMS deployed in Taipei City, we show that the system can achieve a high hit rate in trajectory matching of phone sensed data and the authorized bus data. Moreover, based on the statistics, we demonstrate that the system is capable of ranking the comfort levels of the bus services provided by different agencies, and monitoring the comfort levels of the transportation system overall. The system is also highly scalable without the cost of deploying a sensing infrastructure. We believe that it has the potential to provide a durable and large-scale comfort measuring service for public transportation systems.

1. INTRODUCTION

The *comfort* of rides has been identified as one of the top criteria that affect customers' satisfaction with public transportation systems, and it has been shown that *comfort* is an important consideration for passengers that use public transportation [16, 20, 21]. However, conventional comfort measuring approaches rely on either personal interviews [22] or literature surveys [19], which are generally labor-intensive and time-consuming, and are thus limited in terms of scalability and timeliness.

With recent advances in sensing technologies and mobile handheld devices, *participatory phone sensing* has emerged as a new sensing paradigm that exploits the sensing capabilities of modern smart phones to gather, analyze, and share local knowledge about the mobile phone owners' surroundings [18]. Unlike conventional sensing systems, participatory phone sensing does not rely on dedicated sensing infrastructures and the top-down model of data collection. Actually, it is more penetrative, because it supports *grassroots* sensing (i.e., the bottom-up model), and it encourages participation at personal, social, and urban levels [18].

The concept of participatory phone sensing has been implemented in a variety of real-world applications. For instance, *CenseMe* [28] uses the microphone and accelerometer of smart phones to infers users' activities and social context. Meanwhile, *SoundSense* [27] employs machine learning techniques to classify both general sounds (e.g., music and voices) and discover novel sound events specific to individual users in their daily lives. In [17], Azizyan and Choudhury propose using ambient information (e.g., microphone, camera, accelerometer, and Wi-Fi) to classify the location of a mobile phone. *Nericell* [29] employs mobile smartphones for rich monitoring of road and traffic conditions via an array of sensors (GPS, accelerometer, microphone) and communication radios. Finally, trajectory sensing applications (e.g., *Mobile Google Maps* [8], *Waze* [6], *GeoLife* [7], and *CarWeb* [2]) use the GPS of smart phones to collect users' daily life trajectories and provide different location-based services as an incentive, such as real-time traffic reporting [6, 8] and trajectory recommendation [2, 7]. However, one weakness of these applications is that they are all standalone systems without cross-domain knowledge and cross-system integration. As a consequence, they are limited in their ability to provide value-added services, and they cannot profile large-scale transportation systems as a whole.

In this study, we propose a novel Comfort Measuring System, called CMS, for measuring the comfort levels of rides on public transportation systems. The CMS system is comprised of three parts: 1) data obtained through participatory phone sensing by volunteers who sense and score their daily transportation experiences; 2) the authorized data of public transportation systems, which provides the reliable, accurate, and detailed information about vehicles in the system; and 3) a matching algorithm that mashes up the results of (1) and (2) for further analysis and statistical purposes. Using the VProbe tool [15] and the authorized bus data provided by the Taipei e-bus system [14], we deployed the CMS system in Taipei City. During a 70-day experiment, we collected 425 trajectories, labeled with vehicle identifiers, from 15 volunteers. Based on the results, we make the following observations.

[*]This research was supported in part by the National Science Council of Taiwan under Grants: NSC 97-2628-E-001-007-MY3 and NSC 99-2219-E-001-001.

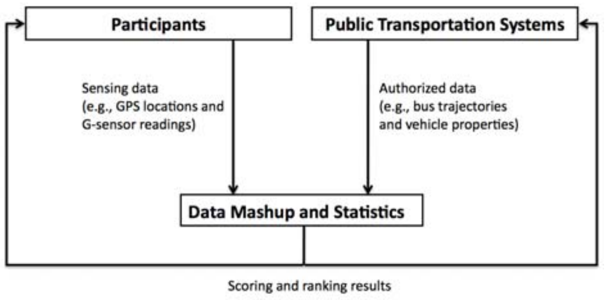

Participants
Public Transportation Systems
Sensing data (e.g., GPS locations and G-sensor readings)
Authorized data (e.g., bus trajectories and vehicle properties)
Data Mashup and Statistics
Scoring and ranking results

Figure 1: The architecture of the CMS system

1. The proposed trajectory matching algorithm can achieve a high hit rate of 93.7%, as long as the contributed sensing data is correct (i.e., without GPS errors) and the trajectory of the vehicle measured is included in the authorized data.

2. In Taipei City's public bus system, 4% of the bus rides are considered comfortable, 17% are uncomfortable, and the rest are in between the two extremes.

3. There is no significant difference in the comfort levels of bus services provided by different bus agencies in Taipei City; and the comfort levels vary a lot among bus services operated by the same agency.

4. Light buses are more uncomfortable than low-floor and the standard (single-decker) buses.

The remainder of this paper is organized as follows. In Section 2, we present the CMS system and a trajectory matching algorithm. In Section 3, we provide a preliminary set of experiment results for the CMS system deployed in Taipei City, and we investigate the factors that affect the comfort levels of public transportation systems in detail. We then summarize our conclusions in Section 4.

2. THE COMFORT MEASURING SYSTEM

In this section, we present the proposed Comfort Measuring System (CMS) for evaluating public transportation systems. CMS is comprised of three components: data collected through *participatory sensing by volunteers, authorized data of public transportation systems*, and *data mashup and statistics*, as shown in Figure 1. We discuss each component in the following subsections.

2.1 Data Collected through Participatory Sensing by Volunteers

The CMS system exploits the capabilities of modern smart phones to sense commuters' transportation experiences in a distributed and participatory manner. CMS does not rely on any particular applications, and it supports many existing smart phone applications that provide raw sensed data about *trajectories* and *vibration measures*, e.g., Dynolicious Log Box [5], MobileLogger [11], SensorLogger [13], Sensor Monitor [12], and VProbe [15].

Specifically, a trajectory is the path of a moving object (i.e., a vehicle) through space. It is usually represented by a set of discrete sample points on the path with a fixed time interval between every two contiguous data points. Each data point contains a timestamp of the sample, and its geographical location information (i.e., the latitude and the longitude).

In addition, the vibration measures contain a sequence of 3-axis accelerations collected by the G-sensor, which is a 3-axis accelerometer now available in most off-the-shelf smart phones. We let $\widetilde{a}_t^x$, $\widetilde{a}_t^y$, and $\widetilde{a}_t^z$ denote, respectively, the accelerations sensed at time t on the X, Y, and Z axes of the smart phone; then we apply the calibration algorithm proposed in [26] to calculate a_t^x, a_t^y, and a_t^z, i.e., the real accelerations at time t on the X, Y, and Z axes fixed to the center of the earth.

We let a_t denote the acceleration of the moving object estimated at time t; and following the ISO 2631 standard [24], we obtain the value of a_t by

$$a_t = \sqrt{(1.4a_t^x)^2 + (1.4a_t^y)^2 + a_t^{z\,2}}, \tag{1}$$

and calculate the *acceleration level* at time t, i.e., L_t, by

$$L_t = 20 \log \frac{a_t}{a_{ref}}, \tag{2}$$

where a_{ref} is a normalization factor with a constant value equal to $10^{-5} m/sec^2$ [25]. Then, following [4], we obtain the *comfort index* at time t, i.e., C_t, by

$$C_t = \begin{cases} 1 & \text{, if } L_t \le 83dB \\ 2 & \text{, if } 83dB < L_t \le 88dB \\ 3 & \text{, if } 88dB < L_t \le 93dB \\ 4 & \text{, if } 93dB < L_t \le 98dB \\ 5 & \text{, if } 98dB < L_t \le 103dB \\ 6 & \text{, if } 103dB < L_t \end{cases} \tag{3}$$

Finally, we calculate the comfort level of a trajectory by averaging all the C_t scores that belong to that trajectory. Intuitively, the smaller the average comfort level, the more comfortable will be the transportation experience.

2.2 Authorized Data of Public Transportation Systems

The CMS system requires the authorized data of public transportation systems, including the vehicle identifiers (i.e., license plate numbers), the vehicles' trajectories, and other miscellaneous information, such as the agency names, the types of the vehicles, and the route numbers. By using the identifier and trajectory information, the CMS associates commuters' sensed data with the authorized data (which we consider in the next subsection), and thereby enables the scoring and ranking of each vehicle in the public transportation system by outsourcing the measurement task to the crowd (i.e., exploiting participatory sensing by volunteers). Moreover, by considering other information about vehicle attributes, CMS can provide more insights into the public transportation system; for example, *Which type of vehicle is most comfortable? Which bus route is most comfortable? Which bus agency provides the most comfortable rides in the city?* In the past, gaining such insights would not have been possible without the deployment of a large-scale infrastructure.

With recent advances in GPS and wireless broadband technologies, an increasing number of major cities world-wide have implemented real-time tracking systems for their public transportation systems, e.g., Boston (MA, USA) [10], Cambridge (UK) [1], Chicago (IL, USA) [3], Seattle (WA, USA) [9], and Taipei (Taiwan) [14]; thus, they provide perfect testing grounds for the CMS system. In this study, we acquired the authorized data of the *Taipei e-bus system* and evaluated the CMS system in Taipei City. However, the CMS system can be applied in any city anywhere, as long as there are people willing to contribute sensing data and the authorized data of the city's public transportation system is available.

The *Taipei e-bus system* was deployed by the Taipei City Government in 2004. In the system, each participating bus has an on-board unit (OBU), which is a thin-client with a GPS receiver and a GPRS interface. The OBU transmits the bus's information (the bus identifier, GPS location, and status codes) to the network control center (NCC) via the GPRS connection periodically (every 15 $\sim$ 25 seconds). In 2010, the e-bus deployment involved 4,028 buses (including low-floor buses, public light buses, and standard single-decker buses) covering 287 routes and 15 operating agencies. The system covers nearly the entire greater Taipei area (i.e., Taipei city, Taipei county, and Keelung City). There are more than 180 million passenger-trips every day. Through our collaboration with the Taipei City government, we were allowed to download real-time bus data every minute. On average, there are 3,865 trajectories and 3,235,460 data points each day.

2.3 Data Mashup and Statistics

In this subsection, we present the trajectory matching algorithm, which finds the most similar trajectory in the authorized data for a given trajectory contributed by a participant. We let TP^k denote the k-th data point of the trajectory logged by the participant's smart phone ($k = 1, 2, \cdots, M$), and let TG_i^j denote the j-th data point of the i-th trajectory in the authorized data ($i = 1, 2, \cdots, G$ and $j = 1, 2, \cdots, N$). Moreover, we define $[TP^k]_i^*$ as the interpolated data point of TP^k on the i-th trajectory in the authorized data (using linear interpolation and based on the timestamp of TP^k).

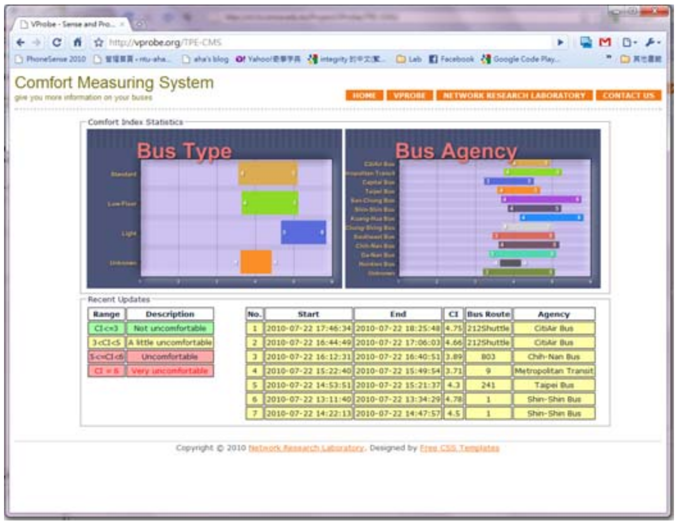
Then, we define Δ_i as the *trajectory distance* between the user-input trajectory and the i-th trajectory of the authorized data. The value of Δ_i is calculated by Equations 4, where $dist(*, *)$ is a distance function that reports the Euclidian distance between the two input GPS locations. Finally, the matching algorithm finds the trajectory $\widetilde{i}$ that has the minimum Δ_i value for $i = 1, \cdots, G$, and regards the $\widetilde{i}$-th trajectory of the authorized data and the user-input trajectory record as the movement of the same vehicle. Thus, the CMS system mashes up the comfort level measurement of the user-input trajectory with the vehicle of the $\widetilde{i}$-th trajectory in the authorized data and manipulates the statistics accordingly.

$$\Delta_i = \sum_{k=1}^{M} dist(TP^k, [TP^k]_i^*) \tag{4}$$

$$\widetilde{i} = \arg\min_i \Delta_i \tag{5}$$

3. PRELIMINARY RESULTS

We now present the preliminary results of the CMS system that we deployed in Taipei City in March 2010. Figure 2 shows a snapshot of the deployed system, called TPE-CMS[1]. Using VProbe [15] as the sensing tool[2], we recruited 15 volunteers to collect bus trajectories in the city. We also asked the volunteers to label each trajectory with the vehicle identifier (i.e., the license plate number), and then used the proposed trajectory matching algorithm to compare the volunteers' labels with the matching results.



Comfort Measuring System
give you more information on your buses

Range	Description
CI<=3	Not uncomfortable
3<CI<5	A little uncomfortable
5<=CI<6	Uncomfortable
CI >= 6	Very uncomfortable

No.	Start	End	CI	Bus Route	Agency
1	2010-07-22 17:46:34	2010-07-22 18:25:48	4.75	212Shuttle	CitiAir Bus
2	2010-07-22 16:44:49	2010-07-22 17:06:03	4.66	212Shuttle	CitiAir Bus
3	2010-07-22 16:12:31	2010-07-22 16:40:51	3.89	803	Chih-Nan Bus
4	2010-07-22 15:22:40	2010-07-22 15:49:54	3.71	9	Metropolitan Transit
5	2010-07-22 14:53:51	2010-07-22 15:21:37	4.3	241	Taipei Bus
6	2010-07-22 13:11:40	2010-07-22 13:34:29	4.78	1	Shin-Shin Bus
7	2010-07-22 14:22:13	2010-07-22 14:47:57	4.5	1	Shin-Shin Bus

Copyright © 2010 Network Research Laboratory. Designed by Free CSS Templates

Figure 2: The screen snapshot of the TPE-CMS system

Table 1: The hit rate of the proposed trajectory matching algorithm

Results	# of trajectories	Percentage
Correct	357	84%
No bus data	41	9.64%
GPS errors	3	0.71%
Miss-matched	24	5.64%

Between March 15 and July 22, 2010, the volunteers contributed a total of 425 trajectories with labels. From the results shown in Table 1, we observe that the proposed matching algorithm can achieve a hit rate of 84% (357/425) in finding the vehicle identifier of the user-input trajectory. Moreover, when analyzing the missed cases, we found that 41 trajectories were missmatched because, according to the authorized data, the vehicles of the labeled trajectories were not in service (i.e., the OBUs were not turned on or they encountered some technical problems). In addition, 3 trajectories were missmatched because there are obvious GPS errors in the user-input trajectories. Thus, after discarding the two cases, our trajectory matching algorithm achieved a hit rate of 93.7% (357/381), which is highly accurate and favorable for the CMS system.

From the results shown in Figure 3, we observe that, among the collected trajectories, only 4% of them were described as comfortable (i.e., $C_t \leq 3.0$), 17% of them were uncomfortable (i.e., $C_t \geq 5.0$), and the rest were in between the two extremes [24]. Moreover, the results in Figure 4 show that the trajectories of bus agency 7 are relatively more uncomfortable than those of the other agencies, while

[1]TPE-CMS: measuring comfort levels of Taipei buses; http://vprobe.org/TPE-CMS/

[2]The sample rates of VProbe are 1 Hz for the GPS and 40 Hz for the G-sensor.

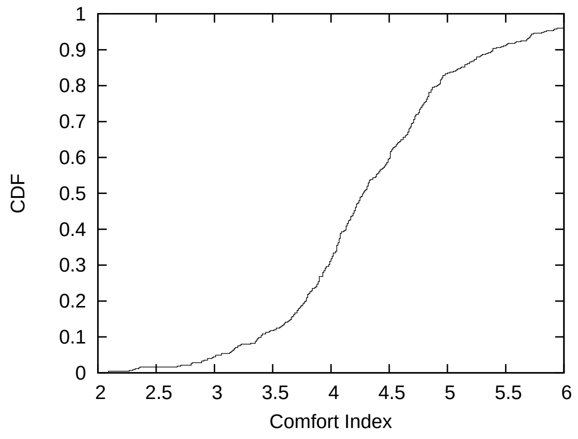
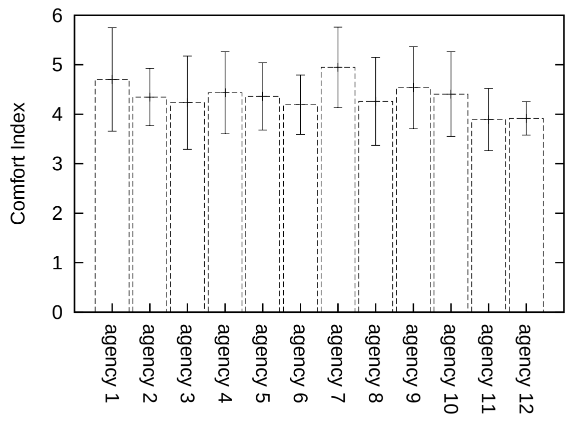
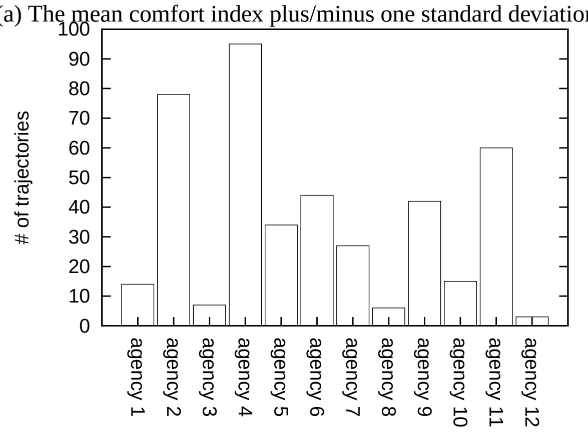


Figure 3: The CDF distribution of the comfort index among the collected trajectories



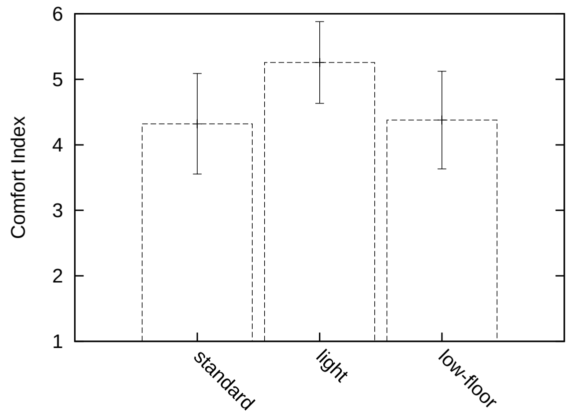
(a) The mean comfort index plus/minus one standard deviation



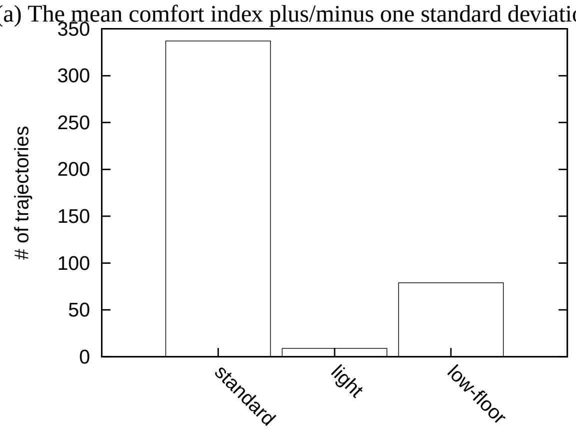
(b) The number of trajectories

Figure 4: The statistics of the collected trajectories based on the bus agencies

the trajectories of bus agencies 11 and 12 are more comfortable. We also observe that the standard deviation of the comfort index is quite large for most bus agencies. The result indicates that the comfort index of trajectories is widely spread, and that 'trajectory diversity' (i.e., the difference in the comfort index across trajectories) does exist within most bus agencies.



(a) The mean comfort index plus/minus one standard deviation



(b) The number of trajectories

Figure 5: The statistics of the collected trajectories based on the bus types

We also investigated the impact of bus types on the comfort indexes of the trajectories. The results in Figure 5 show that the trips on light buses are more uncomfortable than those on low-floor buses and standard buses. This is because the light buses serve suburbs where the routes are usually winding and the roads may not be in prime condition. Interestingly, the trajectories of low-floor buses and standard buses have similar comfort index values, which is counterintuitive to recent reports [23]. The reason is that the low-floor buses serve urban areas; thus, it is inevitable that they will stop more frequently to allow passengers to board/disembark. As a result, there are no significant differences in the comfort indexes of the trajectories of low-floor and standard buses.

4. CONCLUSION

In this paper, we propose a Comfort Measuring System (CMS) for public transportation systems. CMS exploits data collected through participatory phone sensing to measure the comfort level of each vehicle ride. It then mashes up the sensed data with the authorized data of the public transportation system to provide detailed insights into the comfort levels of vehicle rides. Using real data collected from the CMS system deployed in Taipei City, we validate the proposed trajectory matching algorithm, and show that it can achieve a hit rate of 93.7%. Moreover, based on the statistics, we show that only 17% of bus rides in Taipei are considered uncomfortable, and there are no significant differences between different bus agencies.

We also find that the comfort level varies a lot among the bus services provided by the same agency, and smaller buses are the least comfortable vehicles. Work on analyzing other factors that affect comfort levels is ongoing (e.g., road conditions, drivers' behavior, and traffic congestion). We hope to report the results in the near future.

References

[1] Cambridgeshire Bus. http://www.cambridgeshirebus.info/.

[2] CarWeb: A Traffic Data Collection Platform. http://carweb.cs.nctu.edu.tw/.

[3] Chicago Transit Authority. http://www.ctabustracker.com/bustime/home.jsp.

[4] Comfort index table (page 15). http://yctrtrc.ncku.edu.tw/site2/ocwCoursePPT/133trk-slide-2009-III-3.ppt.

[5] Dynolicious Log Box for iPhone. http://itunes.apple.com/us/app/dynolicious-log-box/id307522871.

[6] Free GPS Navigation with Turn by Turn - Waze. http://world.waze.com/.

[7] GeoLife: Building social networks using human location history. http://research.microsoft.com/en-us/projects/geolife/.

[8] Google Maps for mobile. http://www.google.com/mobile/maps/.

[9] ITS Research Program - MyBus. http://www.mybus.org/.

[10] Massachusetts Bay Transportation Authority, T-Tracker Trial. http://www.mbta.com/rider_tools/T-Tracker/.

[11] MobileLogger for iPhone. http://itunes.apple.com/us/app/mobilelogger/id365459773.

[12] Sensor Monitor for iPhone. http://itunes.apple.com/za/app/sensor-monitor/id381075251.

[13] SensorLogger for iPhone. http://amongstbits.com/sensorLogger.html.

[14] Taipei e-bus system. http://www.e-bus.taipei.gov.tw/index.htm.

[15] VProbe: sensing and probing your driving experience. http://vprobe.org/.

[16] S. S. Andaleeb, M. Haq, and R. I. Ahmed. Reforming Innercity Bus Transportation in a Developing Country: A Passenger-Driven Model. *Journal of Public Transportation*, 10(1):1–25, 2007.

[17] M. Azizyan, I. Constandache, and R. R. Choudhury. SurroundSense: mobile phone localization via ambience fingerprinting. In *ACM MobiCom*, 2009.

[18] J. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M. B. Srivastava. Participatory sensing. In *ACM SenSys Workshop on World-Sensor-Web*, 2006.

[19] M. Cantwell, B. Caulfield, and M. O'Mahony. Examining the Factors that Impact Public Transport Commuting Satisfaction. *Journal of Public Transportation*, 12(2):1–21, 2009.

[20] J. Disney. Competing through quality in transport services. *Managing Service Quality*, 8(2):112–118, 1998.

[21] L. Eboli and G. Mazzulla. A New Customer Satisfaction Index for Evaluating Transit Service Quality. *Journal of Public Transportation*, 12:21–37, 2009.

[22] B. Edvardsson. Causes of customer dissatisfaction - Studies of public transport by the critical-incident method. *Managing Service Quality*, 8(3):189–197, 1998.

[23] P. Eriksson and O. Friberg. Ride comfort optimization of a city bus. *Structural and Multidisciplinary Optimization*, 20:67–75, 2000.

[24] ISO. *ISO 2631-1-1997: Mechanical vibration and shock — Evaluation of human exposure to whole-body vibration —*. International Organization for Standardization, Geneva, Switzerland, 1997.

[25] JIS. *Japanese Industrial Standard C 1510 Vibration level meters*. Japanese Standards Association, Tokyo, Japan, 1995.

[26] D. Jurman, M. Jankovec, R. Kamnik, and M. Topic. Calibration and data fusion solution for the miniature attitude and heading reference system. *Sensors and Actuators A: Physical*, 138(2):411–420, 2007.

[27] H. Lu, W. Pan, N. D. Lane, T. Choudhury, and A. T. Campbell. SoundSense: Scalable Sound Sensing for People-Centric Sensing Applications on Mobile Phones. In *ACM/USENIX MobiSys*, 2009.

[28] E. Miluzzo, N. D. Lane, K. Fodor, R. Peterson, H. Lu, M. Musolesi, S. B. Eisenman, X. Zheng, and A. T. Campbell. Sensing meets mobile social networks: the design, implementation and evaluation of the CenceMe application. In *ACM SenSys*, 2008.

[29] P. Mohan, V. N. Padmanabhan, and R. Ramjee. Nericell: Rich Monitoring of Road and Traffic Conditions using Mobile Smartphones. In *ACM SenSys*, 2008.

A Smartphone Based Fall Detector with Online Location Support

Gokhan Remzi Yavuz, Mustafa Eray Kocak, Gokberk Ergun, Hande Alemdar,
Hulya Yalcin, Ozlem Durmaz Incel, Lale Akarun, Cem Ersoy
Computer Engineering Department
Boğaziçi University, Istanbul, Turkey
{remzi.yavuz, eray.kocak, gokberk.ergun, hande.ozgur,
hulya.yalcin, ozlem.durmaz, akarun, ersoy}@boun.edu.tr

ABSTRACT

Falls are identified as a major health risk not only for the elderly but also for people with neurodegenerative diseases, such as epilepsy, and are considered as a major obstacle to independent living. Fast detection of falls would not only decrease the health risks by enabling quick medical response; but also make independent living a safe option for the elderly. In this paper, we propose a fall detector that uses the accelerometers available in smartphones and incorporates different algorithms for robust fall detection such as thresholding and wavelet transforms. We implemented our fall detector on a smart phone running the Android 2 operating system. We performed an extensive set of experiments for evaluating the performance of the implemented fall detector. To the best of our knowledge, although using smartphones for fall detection have been recently studied, evaluating the performance of robust algorithms, rather than thresholding, has not been explored before. Our experimental results show that compared to a simple thresholding algorithm, using wavelet transforms achieve better true positive performance while decreasing the rate of false positives drastically. Besides the fall detection capability, our implementation also provides location information using Google Maps about the person experienced the fall, using the available GPS interface on the smartphone and a warning about the fall and the location information are transmitted to the users, such as the caregivers, via SMS, email and Twitter messages.

1. INTRODUCTION

Falls are risky, especially for the elderly people living independently and people with neurodegenerative diseases. Studies show that, more than one third of the adult population over the age of 65 falls at least once a year in the USA [1]. And up to 30% of these falls result in medium to severe injuries that can lead to the death of the elderly [2]. Besides elderly, patients with neuromotor dysfunction attacks, such as the epilepsy patients, suffer from falls during a seizure due to loss of consciousness. We currently work on a research project together with Istanbul Capa Medical School on monitoring epilepsy patients outdoors and aim to detect seizures that result with a fall event. Quick medical response is desired in fall situations, but the injuries may cause the person to be immobile to the extent that they could not even be able to reach a phone to call for help. One proposed solution to this problem is to use emergency buttons installed throughout the house or on the elderly people themselves so that they can press them in case of a fall related injury. However, if the person ends up in an unconscious state, he/she may not be able to press the button to call for help. Hence, it is important to develop an "automatic fall detection system" that requires no human intervention.

World Health Organization (WHO) defines a fall as an event which results in a person coming to rest inadvertently on the ground or floor or other lower level [3]. Since these events involve motion and change of pose, observing certain characteristics of these may provide us with the necessary information to detect falls. Many types of sensors can be used to observe motion and pose of the elderly and determine if a fall has occurred or not. Current work on automatic fall detection methods can be classified into three main categories in terms of the sensors they use: video-based methods, acoustics-based methods and wearable sensor-based methods [4]. Video-based methods use images provided by cameras installed in the environment and they analyze changes in designated features to detect falls (e.g. orientation and aspect ratio of a bounding ellipse [5]). Acoustics-based methods try to detect falls by detecting vibrations caused by the impact to the ground. For instance, in [6] Zigel et al. propose a method that uses a vibration sensor and a microphone to detect vibrations and noise generated by the impact. Wearable sensor-based methods involve a sensor worn on the subject. For automatic fall detection, methods based on wearable sensors are more attractive since video based methods raise privacy concerns and acoustics based methods are very susceptible to ambient noise. Moreover, video-based and acoustic-based methods require wiring and pre-installation, while wearable sensor based methods will be able to operate as long as the person wears the sensors, even when the user is outdoors.

With the improvements in mobile technology, the cost of smartphones decreased reasonably while their computational capabilities increased. With the decline in the prices, many people currently use smartphones. Many of these smartphones have integrated accelerometers that are used mainly for user interaction and orientation detection. In most of these platforms, it is also possible to access acceleration signals provided by the integrated accelerometer. Such platforms are ideal for developing an application that can automatically detect falls and provide a warning mechanism.

In this paper, we propose a fall detection application which is developed on a Nexus One smartphone running the Android 2.0 operating system. The proposed application uses discrete wavelet transform as a feature extraction method and uses the differences of falls and normal actions in the frequency domain to distinguish them from each other. We evaluate the performance of our fall detection application using 5 different subjects and with a scenario which includes actions that cannot be easily distinguished from a fall, such as jumping or lying on a bed. The following are some of the key contributions and findings of our work:

- To the best of our knowledge, although using smartphones for fall detection have been recently studied [7, 8], evaluating the performance of robust algorithms, rather than thresholding, has not been explored before. We show that, compared to a simple thresholding algorithm, using wavelet transforms achieve better true positive performance while decreasing the rate of false positives drastically.
- We conducted an extensive set of experiments on 5 different users (with different age, height and gender) carrying a smartphone and showed that wavelet transforms achieve 37% better true positive performance.
- Besides automatic fall detection, our application provides location information of the subject using the integrated GPS module and also sends a warning about the fall and the location information to interested users, such as the caregivers, the doctor or ambulance service providers, via SMS, email and Twitter messages.

The rest of the paper is organized as follows: In Section 2, we give a brief overview of the related work. In Section 3, we present details of the proposed application. In Section 4, we explain the experimental setup and present our results. Section 5 draws the conclusions.

2. RELATED WORK

There have been various studies on fall detection using wearable sensors. In [12], Nyan, et al. propose a pre-impact fall detector which uses two sets of sensors, one with an accelerometer and a gyroscope and the other with only an accelerometer. They use the angular data calculated using the sensors to detect falls before the impact to the ground occurs. Although the system yields good lead-time, it is vulnerable to normal activities that cause angular movement on their sensors. Moreover, the need to use two sets of sensors may be uncomfortable for the elderly.

In [13], Chen, et al. propose a fall detector based on a wearable accelerometer. In this work, they assume that a large acceleration impulse will occur upon impact to the ground. When such an impulse is observed on the acceleration signal, they then calculate the orientation of the sensor before and after the impact, and check if there has been a change in the orientation. If both conditions are satisfied, they detect a fall.

Another similar approach described in [11], focuses on acceleration change characteristics during the process of a human body falling. They assume that there are four critical characteristics of a falling event: the *initial status* which is still a non-fall state, *weightlessness* at the start of a fall, *impact* when the human body makes contact with the ground, *motionlessness* since human body cannot rise immediately.

A major criticism to wearable sensor-based approaches is that, the user may forget to charge or wear the sensor, therefore leaving the system in a non-functioning state. With the integration of accelerometers on smartphones, it has become possible to develop fall detector applications that can run on the smartphones. Since people are more likely to carry their phones with themselves, rather than an additional sensor, smartphones with integrated accelerometers can easily be used for pervasive fall detection.

Similar to our work, iFall [7] is an Android application designed for fall detection. However it only uses a basic thresholding method for identifying falls. The algorithm uses two thresholds on the root-sum-of-squares of the accelerometer's three axes. The lower threshold, which is set to 1g, is for identifying the free fall effect and the second threshold, which is set to 3g, is used for capturing the spike occurring when the free fall ends with an impact. This method generates a large number of false alarms; therefore some extra processing is needed. The authors suggest using the orientation of the phone, before and after the impact, in order to validate the fall decisions. Nonetheless, this method still suffers from the similarities in the acceleration signals generated by different actions. PerFallID [8] is another application developed for Android mobile phones. Again, it uses the thresholding mechanism, yet its threshold is adjusted using data collected from real users. However, PerFallID does not have localization support as well as it is lacking warning mechanisms. Therefore in case of a fall, it is unable to inform caregivers or the medical personnel about the fall event and its whereabouts.

Our proposed application is designed to directly address some of these issues and yield better detection result. It uses wavelet decomposition as a feature extraction method, in order to better distinguish falls from non-fall actions. In case of a detected fall, our application can inform predefined caregivers about the event and the location of the event. With the addition of the panic button in the application, the application enables the user to inform caregivers about events that cannot be detected by the application.

3. FALL DETECTOR

Gathering accelerometer signals with different scenarios and with varying system parameters – (including different people doing activities that could potentially be confused with falling, i.e. walking, jumping sitting, lying on the bed) and analyzing these activities in the frequency domain, we realized that the falling activity has very distinctive frequency components that could be useful to distinguish it from the rest of the activities. However, for accelerometer signals whose frequency content varies with time, a simple 1-D Fourier frequency transformation is not sufficient. Although Fourier transform gives what frequency components exist in the signal, it cannot locate the frequencies in the time domain. Short Time Fourier Transform (STFT) is a variation of Fourier Transform, which tries to overcome this problem. In STFT, the signal is divided into windows which can be non-overlapping or sliding and the Fourier transform is applied to these windows. Therefore STFT can localize the frequency components to these windows, but its time-frequency resolution is limited.

Another alternative to Fourier transform, which also give temporal localization for frequency components is the wavelet transformation which has become popular in the last two decades in signal processing due to its ability to give better time-frequency resolution and existence of fast transform algorithms.

3.1 Fall Detection Algorithm

Discrete wavelet transform (DWT) basically yields a multi-scale representation of a discrete signal, formed by iteratively applying the analysis filters to the original signal. The transformation begins with a selection of a mother wavelet, Ψ, from which the analysis filters, h and g, are formed. Then the wavelet coefficients of a discrete signal x at the first scale are calculated as:

$$a_1[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k] = (x * h)[n]$$

$$d_1[n] = \sum_{k=-\infty}^{\infty} x[k]g[n-k] = (x * g)[n]$$

where a represents the approximation coefficients and d represents the detail coefficients. Due to the nature of the analysis filters h and g, approximation and detail coefficients contain half of the frequency components of the original signal and therefore can be subsampled by two. After the subsampling, approximation coefficients can be used to calculate the coefficients for the next scale, effectively forming a filter bank:

$$a_{s+1}[n] = \sum_{k=-\infty}^{\infty} \bar{a}_s[k]h[n-k] = (\bar{a}_s * h)[n]$$

$$d_{s+1}[n] = \sum_{k=-\infty}^{\infty} \bar{a}_s[k]g[n-k] = (\bar{a}_s * g)[n]$$

where $\bar{a}_s$ represents the subsampled approximation coefficients of scale s. The process of the wavelet transformation decomposes a signal by concentrating the signal energy in a relatively small number of coefficients [9]. It is this property of reducing a signal to a comparatively small number of components that makes wavelet-based techniques potentially powerful for signal processing algorithms. A more detailed explanation of DWT can be found in [10].

In our application, we use DWT as a feature extraction method. DWT is applied to the discrete acceleration signal provided by the integrated accelerometer, X, in order to extract the detail coefficients of X at the first scale. Then a predefined threshold, t, is applied to the detail coefficients. If the value of the coefficients is above the threshold, a fall is detected. Formally:

$$F(n) = \begin{cases} 1 \; if \; \overline{d_1}\left(\left\lfloor \frac{n}{2} \right\rfloor\right) > t \\ 0 \; if \; \overline{d_1}\left(\left\lfloor \frac{n}{2} \right\rfloor\right) < t \end{cases}$$

where $F(n)$, is the fall decision for the $X[n]$, and $\overline{d_1}$ is the subsampled detail coefficients at the first scale, and $\lfloor \quad \rfloor$ is the floor function.

3.2 Implementation

The fall detection application is designed for special use of the users who are susceptible to sudden falls like epilepsy patients, the elderly as well as slightly cognitively impaired people, such as the Alzheimer patients. It incorporates the sensing capabilities of the smartphones and sophisticated signal processing techniques to produce a handy application that most people may benefit. The overview of the application is depicted in Figure 1. It basically detects the unexpected fall situations and alerts the caregivers and alternatively the followers of the user in a social network, namely Twitter. The application is also location aware. The emergency messages contain the location information shown on the map. It also has a panic button for general use and an emergency alert cancellation mechanism is available for preventing false alarms.

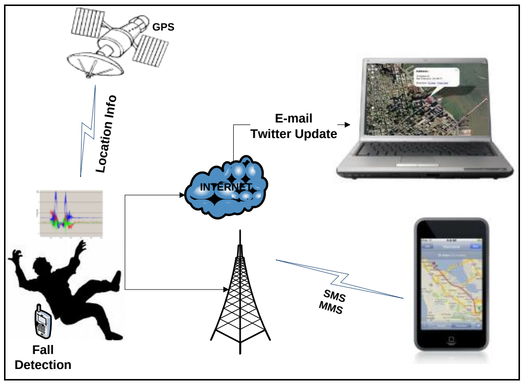


Figure 1. The overview of Fall Detection Application

We implemented our Fall Detector application on the Android 2.0 Platform. Android is an open source mobile operating system and it has a powerful Software Development Kit (SDK) based on Java Framework. It has also SQLite database management system. The core of the application is composed of five parts:

- *Fall Service*: Services are background processes. They are designed to run for long durations and they do not interrupt any other application or process that the mobile phone runs. The "Fall Service" is the most important part of the application, since it allows the fall detection mechanism to run at the background while the user is able to perform other tasks with the phone.
- *Fall Activity*: Activities are the components with which the users directly interact. Activities can be created, started, resumed, paused, stopped, and destroyed. An activity is associated with a user interface, called layout, in the Android application. "Fall Activity" is the visible part of the application and it runs on foreground as depicted in Figure 2.
- *Content Provider*: Content providers are one of the main components of Android applications. All applications can access the data by using a single Content Resolver interface. Content providers not only provide data to the applications but also enable them to share the data among themselves. We use the content provider to access the mobile phone's contact database. In this way, we can select the contacts that will be called when an emergency occurs.
- *Sensor Manager*: "Sensor Manager" allows the application to access the sensors of the mobile phone. We use Sensor Manager to read the acceleration values of the mobile phone's integrated three-axis accelerometer.
- *Location Manager*: By using the "Location Manager", the application is able to obtain the periodic updates for the mobile phone's geographical location retrieved by GPS. In this way, the location of the user is identified when an emergency situation takes place.

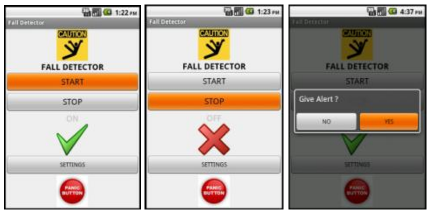


Figure 2. Application Screen Shots

One of the most important features of the application is its simple and efficient interface. There are four large buttons on the main screen. “Start” button starts the fall detection service to run in the background. “Stop” button stops the service. The tick and the cross icons represent the state of the service as being active or inactive. The “Panic Button” at the bottom is used for sending emergency alerts to the caregivers manually. This can be used for when a fall situation is not detected by the service or it may also indicate some other kind of emergency condition, such as a cognitively impaired person getting lost. The location of the user will be delivered to his/her caregivers. The “Settings” button leads the user to the settings screen where he/she can select the contact details of the caregivers and also the user’s account information. There are three types of alerting mechanisms.

1. The user can select the caregivers to be sent SMS message which includes the coordinates of the event. The contact information is taken from the database of the mobile phone.
2. The user can select the caregivers to be sent an e-mail which includes the Google Map link of the fall event. The information is taken from the database of the mobile phone.
3. The user may also select the Twitter update option with his/her own account. In this way, he/she is able to reach more people and one of the closest followers may offer help. Figure 3 shows an example Twitter update.

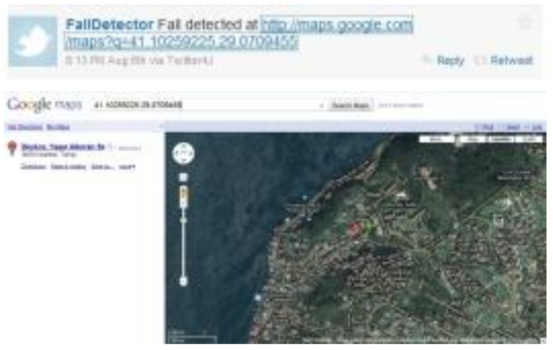

FallDetector Fall detected at http://maps.google.com/maps?q=41.10259225,29.0709455

Figure 3. The Twitter update after a fall, and the Google maps page reached via the link posted in Twitter

When a fall is detected, a notification is displayed together with a sound alert (Figure 4). The users are able to cancel the request within a specific time duration which can be configured. We selected this timeout duration as 20 seconds which is a long-enough period for the people who use the phone relatively slower. If the user did not experience a real fall, he/she can simply cancel the request within the timeout duration. If a real fall has occurred, then the caregivers will be immediately alerted by SMS messages and e-mails together with the social network status update message. This property also helps to solve the problem of differentiating phone falling and user falling.

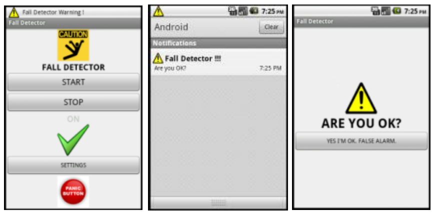


Figure 4. False Alarm Cancellation Mechanism

4. EXPERIMENTAL EVALUATION

For our experiments, we used a Nexus One phone with Android 2.0 operating system installed on it. In the tests, we asked the subjects to repeat a predefined motion scenario, which includes normal actions such as walking, sitting down and lying, and actions that may be challenging to distinguish from falls such as jumping and sitting down quickly.

We conducted the tests on 5 volunteer healthy subjects; each subject repeated the scenario 20 times, yielding a total of 100 sequences which included falls. This data is then processed offline in order to evaluate the performance of the fall detection algorithm using different mother wavelets. Also in order to better evaluate the performance of our proposed method, we implemented the method proposed in [7] for comparisons

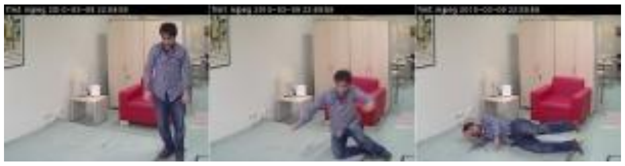


Figure 5. Images of a subject while he is falling, with the phone in his pocket

The performance metrics we use to evaluate the methods are precision and recall. These are defined as:

$$Recall = \frac{TP}{TP + FN} * 100$$

$$Precision = \frac{TP}{TP + FP} * 100$$

where TP is the number of true positives, i.e. falls that are correctly detected by the application, FN is the number of false negatives, i.e. number of falls that could not be detected; FP is the number of false positives, i.e. false alarms. Table 1 shows the recall and precision values of several different mother wavelet selections. The values in the parenthesis either represent the wavelet class, e.g. D12, or the size of the filter generated from the wavelet.

Table 1: Recall and Precision values for different wavelets

Mother Wavelet	Recall (%)	Precision (%)
Daubechies (D12)	88	59
Morlet (N=36)	89	55
Meyer (N=32)	85	95
Gaussian (N=24)	86	46
Biorthogonal (3.5)	90	50

It can be seen in Table 1 that while we get good recall results, half of the fall alarms generated by the application were false alarms in most of the cases. However, we can also see that, use of Meyer wavelet produced 85% recall while retaining 95% precision. The main reason behind these results can be identified in the frequency domain. Figure 6 shows the frequency components of a fall event and the frequency components of a non-fall event (sitting down). As it can be seen from the figure, the acceleration signals generated during a fall have high amplitudes in certain frequencies, while other actions do not. Further analysis show that Meyer wavelet with 32 sample points acts as a band-pass filter for the frequencies at which falls have high amplitudes. Therefore Meyer wavelet can distinguish falls from non-fall actions.

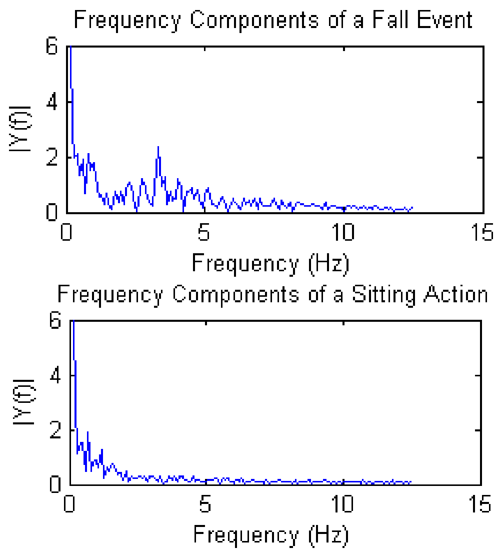


Figure 6. Top: Frequency components of a fall event. Bottom: Frequency components of a sitting action

We also applied the thresholding algorithm proposed by Sposaro, et al. in [7], to the data we have collected. Although we have used their selection of thresholds as a guide, our final threshold selection was made empirically from the data in order to get the best case results. Even in this case, the thresholding method yielded 62% recall and 33% precision. This means that our proposed method performs 37% better than the thresholding method in recall and also has three times more resolution. This also shows that although the underlying assumptions of the thresholding method about the characteristics of a fall event seems reasonable, it suffers from the fact that non-fall events can result in similar acceleration amplitudes. On the other hand, focusing on the frequency components of the acceleration, rather than its amplitude, provides better distinguishing power.

It should be noted here that our scenario includes actions that are less likely to be performed by elderly or the people with neurodegenerative diseases, such as jumping. Therefore, these results can be seen as pessimistic results.

5. CONCLUSIONS

In this paper, we presented a fall detector using the integrated accelerometers on the smartphones, in which we use the discrete wavelet transform as a feature extraction method. As demonstrated by the experimental results, using wavelet transforms yielded significantly better performance, both increasing the true positives by 37% and decreasing the false negatives drastically. Our application not only detects falls, but also provides a location-aware notification service to caregivers and all other interested parties (doctors, ambulance, etc.) through several communication channels, such as GSM, e-mail and Twitter.

In future, we are planning to explore ways of incorporating information from various scales of wavelet transform in order to improve detection performance. We are also planning to conduct experiments in which people are also actually using the phones instead of just carrying the phones on themselves, in order to see the robustness of the application under normal use.

6. ACKNOWLEDGMENTS

This work is supported by the Scientific and Technological Council of Turkey (TUBITAK) under the grant number 108E207, by the Turkish State Planning Organization (DPT) under the TAM Project, number 2007K120610, and by the Bogazici University Research Fund under the grant agreement number "5146".

7. REFERENCES

[1] H. Axer, M. Axer, H. Sauer, O.W. Witte and G. Hagemann, "Falls and gait disorders in geriatric neurology", Clinical Neurology and Neurosurgery, v.112, issue 4, pp. 265-274, 2010

[2] Centers for Disease Control and Prevention, "Falls Among Older Adults: An Overview", http://www.cdc.gov/HomeandRecreationalSafety/Falls/adultfalls.html

[3]http://www.who.int/violence_injury_prevention/other_injury/falls/en/index.html

[4] S. Luo, Q. Hu, "A Dynamic Motion Pattern Analysis Approach to Fall Detection", IEEE International Workshop on Biomedical Circuits and Systems, 2004, p.5-8a-5-8a

[5] H. Alemdar, G. R. Yavuz, M. O. Özen, Y. E. Kara, Ö. D. İncel, L. Akarun, C. Ersoy "A Robust Multimodal Fall Detection Method for Ambient Assisted Living Applications", IEEE 18th Signal Processing and Communications Applications Conference, SIU 2010, Diyarbakır, Turkey, April 2010.

[6] Y. Zigel, D. Litvak, and I. Gannot, "A Method for Automatic Fall Detection of Elderly People Using Floor Vibrations and Sound—Proof of Concept on Human Mimicking Doll Falls", IEEE Transactions on Biomedical Engineering, v. 56, issue 12, pp.2858-2867, 2009

[7] Sposaro F. and Tyson G., iFall: An android application for fall monitoring and response, in Proceedings of IEEE Eng Med Biol Soc, 2009;1:6119-22.

[8] Jiangpeng Dai, Xiaole Bai, Zhimin Yang, Zhaohui Shen, and Dong Xuan, PerFallD: A Pervasive Fall Detection System Using Mobile Phones, in Proceedings of IEEE PerCom Workshop on Pervasive Healthcare (PerHealth), 2010.

[9] I. Daubechies and W. Sweldens, "Factoring wavelet transforms into lifting steps", Journal of Fourier Analysis and Applications, v.4, pp.245–267, 1998

[10] S.Mallat, "A Wavelet Tour of Signal Processing 2nd Edition", Academic Press, 1998

[11] Ning Jia, "Fall Detection Application by Using 3-Axis Accelerometer ADXL345", Analog Devices, Application Note 1023.

[12] M.N. Nyan, F. E.H. Tay, E. Murugasu, "A wearable system for pre-impact fall detection", Journal of Biomechanics, v.41, pp. 3475-3481, 2008

[13] J. Chen, K. Kwong, D. Chang, J. Luk, R. Bajcsy, "Wearable Sensors for Reliable Fall Detection", Engineering in Medicine and Biology 27th Annual Conference, 2005

Did Anybody See That? Smartphone Tracking for Historical Data Retrieval [*]

Vikram P. Munishwar
Computer Science
State University of New York at Binghamton
vmunish1@cs.binghamton.edu

Nael B. Abu-Ghazaleh[†]
School of Computer Science
Carnegie Mellon University, Qatar
naelag@cmu.edu

ABSTRACT

Smartphones have revolutionized the way in which sensing has been performed traditionally. The people-centric nature of smartphone-based sensing enables them to be a part of participatory or opportunistic sensing, where data is collected on a set of designated smartphones and delivered to a server. In this work, we identify the existence of another type of behavior, where the data is not delivered but archived locally on the phones for later retrieval. This type of behavior is common when the phone users capture some data (e.g. a video clip) out of their own interest. However, this complicates the future data retrievals due to the uncontrolled mobility of the data-capturing smartphones. Specifically, the research challenges for later data retrieval include finding the current locations of the required subset of the mobile phones that were present in a specific region at a specific time, without compromising location and identity privacy of the phone user. We discuss existing as well as novel architectural alternatives that can be used to address this problem, along with their qualitative evaluation.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*distributed network, centralized network*; H.3.3 [**Information Storage And Retrieval**]: Information Search And Retrieval—*search process, selection process*

General Terms

Design, Security

Keywords

Smartphone, Indexing, Sensor Network, Network Architecture

1. INTRODUCTION

Recent advances in the processing, communication, and sensing resources on smart phones coupled with their ubiquity and ease-of-use have added new dimensions to the traditional sensing mechanisms [1–4]. For the traditional sensing mechanisms, the focus is on deploying and tasking static or mobile sensors specifically to fulfill the underlying application's requirements. However, with the advent of human-carried devices, such as smartphones, equipped with a number of sensors, such as camera, microphone, accelerometer, magnetometer, gyroscope, etc., the notion of sensing has evolved to introduce many new sensing opportunities. Some of the applications of smartphone based sensing include understanding information about the user's context [14], understanding interactions between people and surroundings [4], and participating in the potentially large-scale active sensing operations [3].

The new sensing mechanisms can be categorized into participatory and opportunistic sensing, depending on the extent of peoples' participation in the sensing activity [6]. Participatory sensing generally involves selecting a group of people to actively participate in sensing useful data for an application [3]. On the other hand, in opportunistic sensing, the person carrying a smartphone does not need to actively participate in the sensing act, but the device itself activates itself at the time of appropriate sensing opportunities defined by an application [5]. While both the approaches differ in their sensing mechanisms, the subsequent data management part is still the same – to deliver the sensed data in real-time or in delay-tolerant fashion to the intended recipient(s).

Although, the real-time data reporting is necessary in a number of applications [7, 10], where the real-time updates can be used as alerts or as feedback mechanisms for actuators, in many occasions the data needs to be stored locally for possible later retrieval. Human-carried phones may store the data locally because: (1) the data is captured out of the phone user's interest, and not because of any underlying task; (2) the importance of the captured data is unknown, since, for instance, the data may be redundant; or (3) the size of the captured data is simply too large to be able to send it, given the energy and bandwidth limitations of the smartphone.

In this work, we focus on the cases where data (or event) is captured by people and stored locally on the smartphone. In such scenarios, it is necessary to send queries to the desired (or target) phones who could have been present at a location of interest at a given time, in order to obtain more information about the event that happened at that location and time. Examples of such location-time specific queries – spatio-temporal queries – include: *Is there a video footage available that was captured immediately before or at the time of an accident, in order to help further investigations?* or simply *How many people were present for yesterday's fire-*

[*]This research work was funded by Qatar National Research Fund (QNRF) under the National Priorities Research Program (NPRP) Grant No.:08-562-1-095

[†]This work was conducted while Dr. Abu-Ghazaleh was on a leave of absense from the State University of New York at Binghamton.

works show? As it can be noted, spatio-temporal queries may involve querying historical data, and thus, it is necessary to understand the current locations of the phones that were present at the event site during the event time.

While, spatio-temporal queries over target/event mobility are supported for the traditional sensor networks [12, 16], they present a significant challenge for people-centric sensor networks due to the uncontrolled mobility of the phone users. Specifically, the challenges include obtaining the set of people who were present in the given spatio-temporal window (at the given location and the time) of a historic or an old event, and deducing their current locations in order to be able to send queries to them. Furthermore, not everyone who was present at the location of the event would have captured the required event data. Thus, it is necessary to reduce the search space by ignoring the mobile phones who may have not captured any useful information. This can enable choosing the k most useful mobile phones for resolving the query. Finally, the data collection operation should maintain the location and identity privacy of the target smartphone user. We discuss these challenges in detail in Section 3.

We present a solution space in terms of the architectural alternatives to address above challenges for efficiently locating the target mobile phones. Specifically, we present and qualitatively evaluate existing architectures as well as a novel semi-centralized architecture, *MobiTrail*, that is based on storing the indexing structure at the intermediate access points. Details of the existing architectures are presented in Section 2, while the solution space for architectural alternatives is presented in Section 4. Finally, concluding remarks and directions for future research are given in Section 5.

2. RELATED WORK

In this section, we present an overview of architectural aspects of existing works related to smartphone networks.

Opportunistic sensing is a form of people-centric sensing, where human-carried mobile phones are tasked to perform sensing activities opportunistically, whenever the application-driven context requirements are fulfilled [4,5]. MetroSense [7] proposes a three-tier architecture for people-centric sensor networks of mobile phones. The lowest-tier consists of mobile phones and static sensors already deployed in the region of interest. The middle-tier comprises of Sensor Access Points (SAPs) deployed separately or integrated with the existing communication infrastructure such as WiFi access points. The roles of a SAP include sensing, tasking or programming sensors, collecting data from them, and providing secure and trusted interactions with them. The upper-tier is a server-tier, where one or multiple Ethernet-connected resource-rich servers act as a core component of the system by providing administrative functionalities. BikeNet [8] also uses opportunistically encountered WiFi access points for delay-tolerant data communication, while the commonly available cellular data channel for real-time data reporting.

Participatory sensing [3] enables creating a community-oriented sensor network that can gather, analyze, and share the local knowledge. It uses pre-deployed WiFi access points or cellular network for data collection purposes. Another form of participatory sensing, Micro-Blog [10], enables smart phone users to upload the sensory data along with location and time information. This data can be geo-tagged to enable viewing the world at a higher resolution. Microblogs are nothing but the user blogs enriched with sensory inputs such as multimedia data associated with location and time. Such microblogs are added to a central database system via the existing WiFi or cellular networks.

MobiSoC [11] presents a middleware that enables mobile social computing applications to help people reconnect with their physical communities and surroundings by leveraging information about peoples' locations, and their social relationships. MobiSoC also presents a centralized architecture, where people can communicate with MobiSoC via the existing WiFi network.

Essentially, almost all of the existing architectures use a centralized approach for data collection, where the sensed data is uploaded to a central unit periodically or opportunistically from the mobile phones. The centrally collected data can in turn be processed/classified and indexed in order to support efficient data-specific query resolution. The similar approach can be used to enable location tracking of mobile phones, which is discussed in Section 4.1

Among distributed approaches, location- and/or time-based query resolution has been studied extensively for traditional wireless sensor networks (WSNs), however the approaches presented are mainly for a network of static nodes. Thus, most of the location-specific indexing approaches are based on the fact that the data about an event happened at a particular location would be present at a sensor located at or nearby that location [12, 16]. Such an approach may not be useful for a smartphone based network, since the phone user may have moved to another location at the time of the query. We discuss the challenges associated with smartphone based networks in the next section.

3. RESEARCH CHALLENGES

The major challenge to locate the target mobile phones that were present in the given spatio-temporal window arises from the fact that the mobile phone (i.e. the data storage) may be moving in uncontrolled mobility pattern. This encourages the need to track the phone, which in turn leads to another problem: preserving identity and location privacy of the mobile phone users. We now discuss these challenges in detail.

Which are the target phones? If the location and the time duration of a historic event is known, then the first step involved in the query resolution process is to determine the set of phones that were present at that location at the given time. For an event that is currently happening, or that has just happened, it is easily possible to probe the access points, if they are available in the event region, which in turn can probe the nearby mobile phones. However, for older events, this approach may not be useful since the phone user may have moved to some other location.

Where are the phones located currently? If the nearby access points stored the information about all the phones that were in its range at different times, it would be possible to address the first challenge. However, it is crucial to understand the current locations of the phones, in order to access the required data from them, which will not be served unless the identity privacy of the phone is compromised.

How to reduce the search space? If some of the phones present at a given location have not captured the required data, they can be discarded from the search space. The third challenge focuses on finding the only subset of phones which may have actually stored the required information, in order to reduce the query resolution cost.

How to maintain location and identity privacy of the phone users? Allowing the system to track the mobile phone users, which were not a part of a particular sensing task or campaign, can raise major privacy concerns. Thus, it is crucial for the system to ensure location and identity privacy of the mobile phones. The privacy requirement negates the possibility of using a trivial solution to learn the current location of the target mobile phone, whereby the event-capturing mobile phone can just report its Id (e.g. its cellular contact number), along with its location and timestamp to a server present on the Internet via a nearby WiFi access point or the cellular data channel. The query-resolution would then just involve locating the mobile phone with the given Id, using the regular cellular infrastructure.

Note that these are the initial set of research challenges stemming from the underlying requirement. However, depending on the solution approach to be used to address this problem, additional set of, mainly systems-specific, requirements can arise, which are out of the scope of this paper.

4. ARCHITECTURAL ALTERNATIVES

Understanding the current location of mobile phones that were present at a given location at given time is crucial to support data retrieval from such nodes. This entails a need to have a suitable architectural support that can enable efficiently locating the target nodes in a scalable manner, without compromising the required privacy of mobile phone user(s). Based on these requirements, we present a design space in terms of possible architectural alternatives that can be used for smartphone-based sensor networks. We categorize the architectures into three types: centralized, semi-centralized, and distributed, depending on where the indexing structure is stored. While the centralized approach is predominantly used in the existing architectures for smartphone networks, the semi-centralized and distributed architectures are hardly explored. We overview the existing centralized architecture, and present a novel semi-centralized architecture in detail. We also suggest a solution approach for employing a distributed architecture for data retrieval.

4.1 Centralized Architecture

In a centralized approach, mobile phones can periodically update their current locations to a central server via nearby data collectors, such as WiFi access points, or by using a cellular data channel [11]. The server can in turn create a spatio-temporal index of the mobile phones' tracking data to figure out the phone that was present in the query-specific spatio-temporal window, and its current location. Infrastructural requirements for such networks are very similar to that of the existing centralized data collection networks [7, 8, 10, 11].

The centralized solutions are beneficial in that the whole indexing structure is available at one location with the information about which phone was located at which place at what time. Thus, it is possible to quickly figure out the set of target mobile phones that need to be contacted for successful query resolution. Furthermore, the server can act as an interface between the smartphone network and the end users, thereby providing a transparent access to the requested data without compromising the location and identity privacy of the target smartphones.

However, the benefits of the centralized approach come at a cost of significant overhead in terms of the bandwidth and energy usage on the mobile phones. First, each phone needs to invest its limited energy supply in performing localization periodically. Localization is a costly operation for a battery-operated mobile phone due to the involvement of radio-communication for it. Furthermore, while the GPS-based localization was observed to provide significantly better accuracy (up to 7 meters), the battery life observed (less than 7 hours) was considerably less than that of the WiFi or cell-towers based localization schemes, when the GPS was used continuously [10]. Second, the estimated location of the phone needs to be communicated via a WiFi access point or a cellular data channel, which again needs to use power-hungry radio communication. Additionally, the granularity of location updates of the mobile phones can significantly affect the overall battery life. Third, potentially all mobile phones that are being used as sensors may be used for this purpose, which will increase the back-haul network's bandwidth utilization considerably, irrespective of the utility of the updates.

4.2 Semi-centralized Architecture: MobiTrail

In order to address the bandwidth and energy-wastage problems with the centralized approaches, we propose a novel semi-centralized approach, *MobiTrail*, to efficiently locate the target mobile phones. The key idea is to store the index at an intermediate (access point) level. Essentially, when a mobile phone senses an event and stores the data locally, it notifies a nearby access point to initialize the trail. We term the access point as a Sensor Access Point (SAP), following the terminology used in the MetroSense Project [7]. The trail can be further maintained by adding entries at the nearby SAPs as the mobile phone moves away from the event location. Since, the SAPs could be associated with the public WiFi access points, it is critical to maintain the location and identity privacy of the mobile phones. Thus, the phone's actual location and its Id (e.g. cellular number) should never be stored at any SAP. If the trail-maintenance cost is prohibitive, techniques such as trail-compaction can be used. Furthermore, techniques to reduce the search space should be introduced to pre-eliminate the phones that do not store the query-specific data. We now discuss the MobiTrail approach in detail.

4.2.1 *Trail initialization*

When a phone senses new data, which can be learned by checking if the corresponding sensor is active or not, its intent behind the sensing activity is verified by checking if it was a part of any assigned task, as in the case of opportunistic sensing. If the sensing activity was not particularly tasked, the newly sensed information can be stored locally. Subsequently, a *trail-initialization* message containing the start and end timestamps, and location of the captured event is sent to a nearby SAP. If an SAP is not available in the nearby region, the notification is delayed until an SAP is encountered during the course of the mobile phone's mobility.

Note that the MobiTrail also needs to localize the phone, to in turn localize the event that it is capturing. However, the localization is performed only when a new event is sensed, and not periodically as in case of the centralized approach.

4.2.2 *Trail maintenance*

The phone's mobility trail is maintained in a similar way to a doubly-linked-list. Specifically, when a new SAP is encountered, the SAP notifies the previous (or preceding) SAP, and adds it as a next hop entry to its routing table to reach the trail-initializing SAP. The previous SAP adds the new SAP as a next hop entry for reaching the target mobile phone. Thus, the phone tracking operation can be performed by first sending the query to an SAP located in the desired event location, and following the trail afterwords to reach the target mobile phone(s). Note that, the query routing process on the SAP-level can take place in a regular manner as a packet routing process on an IP network.

If the phone-user is an active participant in the event capture, further optimizations to reduce the communication for trail-maintenance can be performed. For instance, if the event's importance is progressively decreasing over time, the granularity of trail-maintenance messages can be reduced accordingly.

4.2.3 Trail compaction

The complete trail can be removed if all of the sensed data is delivered to a central server/database as a part of a query response or manual uploading. Trail removal can be performed by traversing the trail backwards from the current location of the phone.

Furthermore, since SAPs are assumed to have a backend connection with the Internet, intermediate SAPs, whose job is to just redirect the query to the next SAP, can be eliminated from the trail. Thus, the compacted trail for each event can only have the first and the last SAP on the trail for that event. Similarly, if a longer trail contains multiple overlapping trails, each for a separate event, then the starting points of all the trails and the ending point must be maintained in the compacted version of the trail.

4.2.4 Reducing the search-space

If a query demands for a specific type of event, the search-space of target mobile phones can be reduced by collecting meta-information about the event at the closest SAP during the trail-initialization phase. The meta-information may include:

1. *Sensors parameters:* Sensors parameters can be used to check if the required sensory input has been captured. For instance, if a query is interested in a video footage of the event, the only mobile phones that had used their cameras can be tracked. Furthermore, if a query is looking for a specific quality of the event capture, other camera parameters such as the capturing-resolution, camera zoom, etc. can be used to deduce the event coverage quality.

2. *Event summaries:* The search space can be further reduced by supplying event summaries (e.g. wavelet based summaries [9]) to the nearest SAP during the trail-initialization phase. These summaries can be used to perform light-weight pre-matching for the query, to select and track the best matching mobile phones. For instance, all the mobile phones, who have captured the event data, may not be required in many cases, and thus the search-space reduction mechanism can be used to select the k most matching mobile phones to resolve the query.

4.2.5 Location and Identity Privacy

The fact that a phone is being tracked without having appropriate permissions from the phone user, may raise privacy concerns for the phone users. Thus, it is crucial for the system to ensure location and identity privacy of the mobile phones that it will track. Since, the query will be specific to the event-location, the actual location of the mobile phone capturing the event will never be exposed to the end users. For query resolution, the query needs to be sent to a SAP located near the event region, and never to the mobile phone who has captured the event data. Once the query reaches the SAP present in the query-specific region, it can follow the trail to reach the last SAP that can access the required data from the target mobile phone present in its communication range. If the target phone is not in the communication range of the last SAP on the trail, it needs to wait until it receives a message from the next SAP about the phone's presence in its communication range.

In order to maintain identity privacy, we propose to use event-specific identifiers, instead of phone-specific identifiers. Essentially, in the trail initialization phase, the SAP generates a unique event-id for the target phone, and uses it in its routing table, and also notifies it to the target phone. The event-id needs to be unique only for the SAP that generates it. The further SAPs on the trail utilize the same event-id, if it has not been used by them already. Otherwise, the SAP generates a new unique Id (unique specific to itself), and stores a mapping from the old Id to the new Id. If the mobile phone captures more than one events while moving, the corresponding SAPs store a mapping from multiple event-ids to a single unique event-id, and assign the new event-id to the phone.

It is critical to note that the unique Id generation and maintenance is a difficult, if not impossible, task in case of the centralized solution, because the Ids are event-specific and not phone-specific. Since, each phone can participate in sensing multiple events, potentially huge number of unique Ids need to be generated and maintained consistently at a global scale for a centralized scheme. MobiTrail avoids this problem by making use of the geographically distributed nature of the smart phones and SAPs. Thus, it is significantly simpler to maintain uniqueness at an individual SAP level, since the number of phones and the number of events that can potentially be in a range of a SAP could be very limited.

Although, data privacy is not a focus of this paper, user level control for tagging data as public or private coupled with traditional access control mechanisms can be explored [13].

4.2.6 Summary

In summary, MobiTrail provides a semi-centralized way to locate phones in a given spatio-temporal window. The advantages of MobiTrail are that it is a scalable, and privacy-preserving approach. In addition, it also supports optimizing the query resolution task by using the meta-information about the desired event, in terms of sensor parameters or event-data summaries, to pre-classify the most useful target mobile phones. Finally, the identity privacy ensuring mechanism is considerably easier than that in the centralized approach, which is based on exploiting the geographical separation of SAPs. The disadvantage of MobiTrail is that it has a slight overhead of maintaining the indexing structure at the SAP level, and the phone tracking process may take a little longer than the centralized approach.

4.3 Distributed Architecture

In a distributed architecture, indexing structure is stored directly on the mobile phones. The target mobile phone search problem is similar in nature with the search problem in peer-to-peer (P2P) networks. Thus, a distributed P2P lookup approach, such as Chord [15], can be adapted to solve the target mobile phone lookup problem, based on a given key representing the query-specified spatio-temporal window.

Note that the idea for distributed approach discussed here is in a preliminary stage, and needs more work to show its accuracy, while maintaining scalability and robustness for mobile phones case, which is a part of our future work. Advantages of using a distributed approach include scalability, and no need for the infrastructure to be in place to support query resolution. The disadvantages are more implementation-specific, and may include the lack of identity privacy for the phone users, and a possibly longer query resolution time in comparison with the centralized approach.

5. CONCLUSION AND FUTURE WORK

Smartphone based sensing is being used widely either as a part of the assigned sensing task, or simply out of the user's own interest. Majority of the existing work focused on sensing data as a part of the assigned task, and reporting it in real-time or opportunistically to the intended receiver(s). In this work, we focused on the cases where the sensed data is stored locally on the mobile phones. We identified a problem of efficiently locating the subset of mobile phones that have potentially captured data about the historical event of interest. We discussed research challenges associated with this problem, and presented a design space aimed at addressing the challenges. The design space provided a categorization of the architectural alternatives, including a novel semi-centralized approach, *MobiTrail*, to address this problem, along with their qualitative evaluations.

In future, we plan to quantitatively evaluate MobiTrail in simulations as well as on a testbed comprising of smartphones and SAPs. In addition, we plan to work on developing the distributed indexing structure for smartphones, whose solution approach is briefly discussed in the paper.

6. REFERENCES

[1] I. Akyildiz, T. Melodia, and K. Chowdhury. A survey on wireless multimedia sensor networks. *Computer Networks*, 51(4):921–960, 2007.

[2] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer networks*, 38(4):393–422, 2002.

[3] J. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M. Srivastava. Participatory sensing. In *World Sensor Web Workshop*, pages 1–5. Citeseer, 2006.

[4] A. Campbell, S. Eisenman, N. Lane, E. Miluzzo, and R. Peterson. People-centric urban sensing. In *Proceedings of the 2nd annual international workshop on Wireless internet*, page 18. ACM, 2006.

[5] A. Campbell, S. Eisenman, N. Lane, E. Miluzzo, R. Peterson, H. Lu, X. Zheng, M. Musolesi, et al. The rise of people-centric sensing. *IEEE Internet Computing*, pages 12–21, 2008.

[6] S. Eisenman. *People-centric mobile sensing networks.* PhD thesis, Citeseer, 2008.

[7] S. Eisenman, N. Lane, E. Miluzzo, R. Peterson, G. Ahn, and A. Campbell. Metrosense project: People-centric sensing at scale. In *First Workshop on World-Sensor-Web (WSW.2006).* Citeseer, 2006.

[8] S. Eisenman, E. Miluzzo, N. Lane, R. Peterson, G. Ahn, and A. Campbell. BikeNet: A mobile sensing system for cyclist experience mapping. *ACM Transactions on Sensor Networks (TOSN)*, 6(1), 2009.

[9] D. Ganesan, D. Perelyubskiy, and J. Heidemann. An evaluation of multi-resolution storage for sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 89–102. ACM Press New York, NY, USA, 2003.

[10] S. Gaonkar, J. Li, R. Choudhury, L. Cox, and A. Schmidt. Micro-blog: Sharing and querying content through mobile phones and social participation. In *Proceeding of the 6th international conference on Mobile systems, applications, and services*, pages 174–186. ACM, 2008.

[11] A. Gupta, A. Kalra, D. Boston, and C. Borcea. MobiSoC: a middleware for mobile social computing applications. *Mobile Networks and Applications*, 14(1):35–52, 2009.

[12] X. Li, Y. Kim, R. Govindan, and W. Hong. Multi-dimensional range queries in sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 63–75. ACM New York, NY, USA, 2003.

[13] Q. Liu, R. Safavi-Naini, and N. Sheppard. Digital rights management for content distribution. In *Proceedings of the Australasian information security workshop conference on ACSW frontiers 2003-Volume 21.* Australian Computer Society, Inc., 2003.

[14] M. Raento, A. Oulasvirta, R. Petit, and H. Toivonen. ContextPhone: A prototyping platform for context-aware mobile applications. *IEEE Pervasive Computing*, pages 51–59, 2005.

[15] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, page 160. ACM, 2001.

[16] J. Winter and W. Lee. KPT: a dynamic KNN query processing algorithm for location-aware sensor networks. In *Proceeedings of the 1st international workshop on Data management for sensor networks: in conjunction with VLDB 2004*, pages 119–124. ACM, 2004.

NRC Data Collection and the Privacy by Design Principles

Imad Aad and Valtteri Niemi
Nokia Research Center, Lausanne
firstname.lastname@nokia.com

ABSTRACT

Nokia Research Center (NRC) in Lausanne, Switzerland has launched a rich data collection campaign during fall 2009, the purpose of which is to study user socio-geographical behavior, mobility patterns etc. of approximately 200 people. All sensors on the mobile devices (GPS, microphone, wireless interfaces etc.) were frequently activated in order to grab the most of the user contexts, to be generic enough and answer the needs of various researchers using the datasets.

The data is as rich as it could be without being too intrusive into the volunteers' lives. We therefore took particular care in preserving the privacy of the participants, while still keeping the data useful for the various future analyses. In this paper we describe the anonymization techniques that we applied to the data, how we met the principles of privacy by design, and the legal aspects with the participants and the researchers.

1. INTRODUCTION

In order to study user behavior, mobility, social interactions etc. NRC launched a data collection campaign in fall 2009 [1]. A number of volunteers were given high-end Nokia phones, equipped with a special software client capable of gathering rich data collected from the phone sensors 24/24 hours, 7/7 days, in order to get a deep insight of user activities. The logged data is automatically uploaded every day to a database server, anonymized, before being accessible to researchers.

The collected data includes GPS coordinates (in beginning of October 2010 we have around 9 million GPS entries in total), acceleration data(1 million samples), surrounding WLANs (458K unique WiFi access points; 43M access points seen in total), BlueTooth devices (414K unique BT addresses; 26M BT encounters), GSM cells (83K unique cell towers, 25K in Switzerland; 39M towers seen in total), incoming and outgoing call (327K calls) and SMS numbers (146K text messages), contacts list (99K phone book entries), calendar/memo, as well as the media played or being recorded (including 47K pictures/videos taken and 120K songs played). About half of the volunteers opted for sampling the audio features (497K audio samples). Altogether, we have almost 178M entries in the data base, 194 participants who have visited 62 countries and collected around one thousand person months of 24/7 data (as of beginning of October 2010).

On the other hand, no content of communication (call, sms), documents, nor URLs is recorded.

The research interests of the users of the (anonymized) data base cover a very wide range: from sociologists, to mobility modelers, socio-geographical analyzers, networking, security and privacy researchers. The number of new data users and new areas of interest have been increasing steadily since the launch of the campaign.

This richness in context data raises many privacy issues, and defining the anonymization techniques becomes challenging when it comes to keeping the data as useful as possible, even for future (not yet specified) research purposes. The details of our anonymization techniques, their compliance to the Privacy by Design principles [2], the encountered compromises to be made, and the envisioned improvements are the main contents of this paper.

The seven principles of "Privacy by Design" (PbD) [2] are:

1. *Proactive* not Reactive; *Preventative* not Remedial
2. Privacy as the *Default*
3. Privacy *Embedded* into Design
4. Full Functionality - Positive-Sum, not Zero-Sum
5. End-to-End Lifecycle Protection
6. Visibility and Transparency
7. Respect for User Privacy

These principles provide an excellent framework for system design, especially when privacy-sensitive data is involved. For most of the principles, the name of the principle gives already a pretty good idea about what is meant. The principle of "Positive-Sum" captures the important goal that privacy protection should not have negative impacts on other properties of the system, e.g. usability or performance. The last principle "Respect for User Privacy" can be seen almost like a meta-level principle: applying all other principles from user point of view gives a good basis for complying with the seventh principle as well. More detailed descriptions of the seven principles can be found in [2].

In this paper, we use the framework of PbD principles to analyze how user privacy have been taken into account in the NRC Lausanne data collection campaign [1]. Section 2 contains a brief discussion of two facets: first, the positioning of our data set in the vast amount of various data sets used for research purposes, and secondly, challenges that are faced when anonymization techniques are used with privacy-sensitive data sets. In Section 3 we explain details of our anonymization methods that are in use for the collected data, and we also discuss some legal aspects relevant for our campaign. Section 4 is organized according to the seven PbD principles, and we discuss how each principle has been followed in the data collection. In Section 5, we present some enhancements to our anonymization mechanisms. These have not yet been put into use but could in principle be applicable to the present set of collected data. Finally we give some concluding remarks in Section 6.

2. BACKGROUND

Driven by the increasing success of social networking, the various businesses behind it, and the increasing capabilities of smart-phone capabilities in sensing user context [7], many research groups, operators, and ISPs are now exploring the potentials of mining rich context data [9, 6]. Even the general public begins to be aware of the astronomical amounts of data that exist in various data bases.

Collecting and opening such data bases for research purposes definitely comes with considerable work on anonymizing it in order to preserve the user privacy [5] since it often comes without the users consent, especially when it is a large scale data such the ones done by phone operators or ISPs on their clients [8].

Many of the existing data collection campaigns have specific focuses like collecting phone usage statistics, identifying groups of people, human behavior etc. separately. In contrast to those, the data collected in our campaign is as generic as it can be, limited only by the (already high) sensing capacities of the phones deployed. From the users side, no specific research group or research interest was predefined, leaving it open potentially to many groups of researchers with various topics of interest. More details on our data collection can be found in [1].

Such rich data, even when anonymized, typically leave some identifiers easily usable to trace back the identities of users allowing security or privacy researchers to get credit in deanonymizing them [10], nevertheless resulting in privacy scandals such as [3].

Building upon the experiences learned from others, and to enforce privacy-preservation of the users, technical and legal techniques were put in place for our campaign. Throughout the data anonymization work we used legacy anonymization techniques and primitives, without having to design new ones. The challenging part, however, turned out to be the degree to which we should apply anonymization, and still keep the data useful.

All individual anonymization techniques used here such as keyed hashing, coordinate truncation etc. can be commonly found in the literature [5], and are used to anonymize individual databases. In our case the rich set of data types imposed specific combinations of anonymization primitives, applied to a certain degree, in order to keep the data useful. To fill the gap between usable and perfect anonymization, legal agreements are done with the researchers, mainly because of protecting the users' privacy.

3. ANONYMIZING PIs AND PIIs

In this section we describe our anonymization approach for Personal Information (PI) and Personally Identifiable Information (PII) in the data base.

3.1 What is anonymized and how

With relevance to anonymization, the collected data can be treated as three different types: GPS coordinates that get truncated, textual data that get hashed, and acoustic data that get sampled and shuffled.

By "hashing" an info we mean concatenating the message as (Key1||info||Key2) then hashing it using SHA256 function. The use of a hash function provides one-way property: it is infeasible to compute "info" from the hashed version. This is a keyed hash construction; without access to the keys it is neither possible to check whether a specific "info" (obtained, e.g., by an educated guess) leads to a given hashed version. We did not have the need to use a (slightly more complex) HMAC construction [12] because we have a fairly restricted range of use for the function and the data format and length of all entries in the data base is well understood and controlled.

"Info" is converted to lowercase beforehand. Note that this keeps the data consistent, in the sense that data entries that have differences only for the case will anyway result in equal hashes. For implementation of these hash functions we used an SQL library called pgcrypto.sql [11].

GPS coordinates are stored with three different precision levels; we give each research group access to the one that is sufficient for their purposes. The different precisions levels are: complete GPS coordinates, removing the last 2 digits and rounding (which, in Switzerland, results in an accuracy of around 110 m in latitude and 80 m in longitude), removing the last 3 digits and rounding (accuracy of roughly 1 km for Switzerland). The truncated coordinates result in step-like paths which increase the ambiguity level. The resulting ambiguity level depends on the initial geographical area: in rural areas, the step-like paths can be easily mapped back to the (only?) road, and the path ends to the (only?) house. Whereas in dense city centers such truncation results in high ambiguity levels, proportional to the number of streets/flats within the output path "step". An adaptive approach is discussed in Section 5.

Phone numbers (in phonebooks and caller/callee lists) have the last 7 digits hashed, while the first ones are kept in clear. Such prefixes are useful to identify the regions and to distinguish mobile phone numbers from landline ones. All names (of users, contacts in contacts list, caller, callee etc.) are hashed

MAC addresses (of WLAN, BlueTooth devices) have their last 6 digits hashed. The first 6 are left in clear text since they point to the chip manufacturer etc. This provides still a high ambiguity about the user ID while indicating, for instance, what kind of devices are in the neighborhood.

SSIDs of WLANs are hashed, since it is common practice that families or companies set their wireless network SSIDs similar to their own names.

Other data such as calendar titles and location (text), filenames of media generated (e.g. pictures), names of folders (Boxes) for text messages are entirely hashed since they are typically personalized, therefore likely to reveal PIIs. Phone IMEIs (i.e. serial numbers) are also entirely hashed.

Acoustic data are recorded in order to help identifying various environments (e.g. noisy, quiet...) of individual users, or to distinguish between different locations/rooms of different users in geographical proximity. This data is read every 10 minutes for a duration of 30 seconds, utilizing Mel-Frequency Cepstral Coefficients (MFCC) [13]. These same coefficients are typically used for speech recognizers and they do not provide alone high enough privacy. In order to increase the privacy level, we randomly shuffle the time order of the individual parts so that the content or identity of the speaker can not be detected anymore. In contrast with the other data types that get anonymized after upload onto the data base, acoustic data is scrambled (therefore anonymized) on the user device itself prior to the upload. After randomization, certain statistical properties of the acoustic sample are still preserved, and they are sufficient to provide information about the environment.

3.2 What is not anonymized

Names of media played (music, album, track number etc.) are kept in clear text since it is valuable for user profiling only when kept so, and these reveal no privacy-sensitive information in practice. Hashing such information would imply big losses in contextual data for negligible improvements in privacy.

Cell tower IDs that the mobile sees are kept in clear, and so is the level of received signal power. Other local system data such as battery status and level, running application(s), screensavers etc. is kept in clear.

No transformation is applied to acceleration data.

3.3 Legal commitments from the researchers, to the participants

As one may infer from the previous description of the techniques, single data types provide little personal information about the participants, but reverse-engineering gets easier when more data types are combined in order to reveal the participant identities or private information.

In order to complement the technical anonymization functions, researchers are tied with legal commitments not to reverse-engineer the data and keep the participants' privacy preserved. Note that unlike many other user data bases that grant access to any user, accessing NRC data collection is tied to "Data Sharing Agreements" between the research institution, the individual users, and Nokia. Prior to granting access, the purpose of the research is discussed, and the commitments on preserving participants privacy is made clear, then the legal agreements are finalized. Of course, these restrictions are somewhat unfortunate because they restrict the open access to the data set that would of course be beneficial to the scientific community.

On the other hand, prior to filling and signing the consent forms, participants were carefully informed about the research targets, data collection, storage, transfer, and anonymization methodologies used, as well as the awaited benefits. Furthermore they were trained on how to visualize their (raw and statistical) data, share it with friends, or how to delete it, using a dedicated and easy to use web page. Regular events took place, during which the participants were updated with the above information, the campaign news and statistics.

The participants have the right to leave the campaign at any time (which a few of them did, mainly due to the short battery lifetime or because of leaving the country.) Nokia reserved the right to exclude participants from the campaign in the event of non-compliance with the protocol, which never happened so far.

3.4 The compromises during anonymization

In this section we discuss various compromises done in order to find the right balance between anonymity and utility. The two extremes can be briefly described by the two examples:

- High anonymity levels could be achieved by removing all kinds of identifiers from the GPS coordinates, call logs etc. This would make it hard to construct e.g. paths, and therefore reverse engineering of the identities would also be hard to perform. However, this would drastically reduce the linkability between calls, events, coordinates etc. hence degrading the usability for mobility models, socio-geographical analysis etc.
- High utility and usability levels could be achieved by leaving contextual data in clear text. Indeed, this would put the data into a perfect shape for context analysis (e.g. social interactions). However, reverse engineering would become an easy task for finding people's identities and their whereabouts.

To avoid drawbacks illustrated by the above examples, some subtle compromises were to be done for anonymizing GPS coordinates, and many textual data that can be used as PIIs.

GPS coordinates

One easy option to strongly anonymize the GPS coordinates would be to (key-) hash them, similar to what was done for textual data. This still provides identical outputs for identical inputs, while securing against reverse-engineering of the private locations / IDs. Apart from preserving the "sameness" property, hashing results in "randomized" coordinates: inputting a user's path would output random geographical jumps, therefore losing the information about speed, proximity, and visited Points of Interests (PoI).

Another option where speed and proximity can be preserved is the use of linear transformation of the GPS coordinates: a given path input to the anonymization function results in a translated/rotated/scaled path geographically distant or different from the initial one, therefore anonymizing "private" coordinates. This approach has the following drawbacks:

- Most of the movements of people are along roads and highways rather than arbitrary paths in forests/lakes etc. Those road shapes are easily identifiable, often even visually, in the output space, hence making them easy to reverse and map to the original coordinates.
- Transforming a set of coordinates into another, located somewhere else on the globe removes all information of PoIs, which is a useful information component for many research areas. For instance, a user going from work, to a bar, then to a cinema, then to home obviously has a different profile from one going from point A, to B, to C then D in the middle of the pacific ocean.

Truncating the GPS coordinates provides a good (and very simple) balance between anonymity and usability. Public PoIs can be identified as such, then tagged, and stored in the database.[1]

Textual data

Leaving the data in clear text easily could make the data base users, intentionally or not, break the privacy of the participants. On the other hand randomizing it makes it completely useless. The adopted hashing technique preserves sameness and the resulting data showed to be still highly useful to most researchers. However, the similarity between calendar entries *"meeting with John"* and *"meeting with Laura"* is lost after hashing, and "meetings" are not identifiable neither. Advanced anonymization techniques that tackle this problem are discussed in Section 5.

Researchers who are also participants

Another type of compromise is encountered because of the fact that quite many (although only a small minority of) campaign participants are EPFL staff/students, among whom several are also data base users at the same time. These persons have access to their own clear text data on their phones or over the web interface, and they also have access to the same data in anonymized form in their role as a researcher. In principle, this enables them to easily create a mapping between certain data items and their anonymized counterparts. As explained in the previous subsection, their role as a researcher prevents them from doing such reverse-engineering (by contractual means).

The issue is not quite as simple, though. For many research topics, such a mapping would be quite useful. An example is the name of a static WLAN access point. On the other hand, de-anonymization of such a static AP in a public place leads to a minimal privacy violation because, similarly, proximity to public PoIs is visible in the data base. To avoid situations where a researcher has a temptation to try de-anonymization for the purpose of progress in his research work, a reverse table that maps the anonymized data items back to the original form is provided to the researchers in cases where it is shown that no privacy violations are introduced because of this. In particular, for the case of WLAN APs, such APs located at the campus of EPFL can still be identified with their SSIDs, because a reverse table is provided to the researchers.

[1]Work in progress.

4. HOW WELL WE MEET THE PbD PRINCIPLES?

1. Proactive not reactive: the most important point is that all privacy-sensitive data is indeed anonymized. Because we have wanted to impose minimal restrictions to the nature of the research problems that could be addressed using the collected data, we have tried to avoid anonymizing too much. On the other hand, this kind of "future-proofing" has implied that breaking the anonymization is possible for a skillful person with sufficient amount of local knowledge about Lausanne and its inhabitants. Therefore, we have been forced to use legal type of protection against reverse-engineering: data access is only provided for researchers who commit themselves to NOT trying to break the anonymization. This is also the main reason why we cannot release the full data set to completely public usage.

2. Privacy as the Default: anonymization is indeed automatically enabled all the time. As explained in the previous section, there are various levels of anonymization, especially for location. The default level of anonymization is always the strongest and could be relaxed if the research problem necessitates it.

3. Privacy Embedded into Design: anonymization is a key feature of the system architecture and the whole campaign design.

4. Full Functionality: anonymized data is sufficient for research purposes but we cannot exclude the possibility that some valuable research opportunities are lost because of it. For instance, providing content of communication in the data base would certainly have opened many new vistas for studying users' contexts (and in general much better view on the social life of campaign participants).

5. End-to-End Lifecycle: anonymization and access control will be enforced throughout the lifetime of the data base.

6. Visibility and Transparency: users have full view to their own data and they are able to delete anything they want. On the other hand, individual accesses to the anonymized data by researchers is not visible to individual participants of the data collection campaign. In principle that kind of transparency could have been arranged also but it is hard to see what kind of purpose it would serve.

7. Respect User Privacy: privacy has been the key element in the whole campaign design; the data is (also) used for creating better privacy protection mechanisms.

5. CAN WE DO BETTER ANONYMIZATION?

In this section we give a couple of examples about how our anonymization techniques could be enhanced while still keeping them applicable to our setting. Certainly, there are also other possible directions for improvements, but in this paper we focus only on these examples.

Regarding GPS coordinates, the future improvement step could be to adapt the truncation of GPS coordinates to the densities of roads, houses, and population of the various visited areas, so as to maintain a constant level of ambiguity, regardless of whether the area is rural or densely populated. This, however, requires a rough knowledge of demographics of the visited areas.

A more challenging improvement is the one for hashing textual data as hinted in Section 3. So far entire data entries (e.g. "meeting with John") were hashed, therefore if another "meeting with Bob" shows in the calendar, or "John's Birthday", nothing can be found in common after anonymization: "meetings" are hard to identify in the agendas, and so is "John" if his name appears among another text. In order to improve this and increase utility for researchers, while maintaining equal levels of privacy, hashing individual words seems to be appropriate and the hash("John") can be identified in the anonymized data base, whether for "meeting" or "Birthday". This comes, however, with several challenges:

- Sentences (i.e. data entries) should not have common links because of hashes of common words like "with", "and", "or" etc. Therefore such publicly common words should be kept in clear, or a dictionary of hashes of common words is written (somehow in analogy with tagging public PoIs in the truncated GPS coordinates). This task requires natural language processing techniques, applied to the various languages spoken by people in the campaign (at least six languages).
- Even the same word ("meeting ...") may have different meanings in different contexts such as in business or social life. Same word in different contexts results in different privacy-sensitivities, and therefore anonymization techniques should be applied accordingly.
- Especially in the extreme case where every word is hashed separately we would run into the problem well known with so-called Electronic Code Book (ECB) mode of encryption. Indeed, in a large data base, it would be easy to distinguish more commonly used words from more rarely occurring words. Furthermore, after every new word that is inverted/decrypted, the task of inverting/decrypting the rest becomes easier.

These improvements would clearly help in better understanding of user context (increasing utility of the data set) without degrading anonymity. However, it comes at the cost of designing proper natural language processing techniques and applying it to the existing data.

6. CONCLUSION

We showed how we anonymized the data of our rich data collection campaign. The data set is highly privacy-sensitive and therefore privacy protection is needed. Because there is a wide range of research areas that could potentially utilize our rich data, the anonymization and utility have been carefully balanced. Some legal counter-measures were also needed against reverse-engineering efforts. We also discussed potential enhancements to the current anonymization techniques.

We hope our findings and approach can be useful for other researchers anonymizing other collected data.

7. REFERENCES

[1] N. Kiukkonen, J. Blom, O. Dousse, D. Gatica-Perez and J. Laurila, "'Towards rich mobile phone datasets: Lausanne data collection campaign", in Proceedings of ICPS 2010.

[2] http://www.privacybydesign.ca

[3] http://en.wikipedia.org/wiki/AOL_search_data_scandal

[4] Barbaro, M., Zeller Jr., T.: A face is exposed for AOL searcher no. 4417749. New York Times (August 9, 2006), http:www.nytimes.com20060809technology09aol.html

[5] Narayanan, A., Shmatikov, V.: Robust de-anonymization of large sparse datasets. In: Proceedings of the 2008 IEEE Symposium on Security and Privacy. pp. 111-125 (2008)

[6] Hossein Falaki, Ratul Mahajan, Srikanth Kandula, Dimitrios Lymberopoulos, Ramesh Govindan, and Deborah Estrin, Diversity in Smartphone Usage, in MobiSys'10, June 2010

[7] N. Eagle (2010), "Mobile Phones as Social Sensors", The Handbook of Emergent Technologies in Social Research, Oxford University Press (in press).

[8] J. Blumenstock, D. Gillick, and N. Eagle (2010), "Who's Calling? Demographics of Mobile Phone Use in Rwanda", AAAI Spring Symposium 2010 on Artifical Intelligence for Development (AI-D) (in press)

[9] http://reality.media.mit.edu/

[10] A. Narayanan, V. Shmatikov, De-anonymizing Social Networks, in Proceedings of S&P 2009

[11] http://anoncvs.postgresql.org/cvsweb.cgi/pgsql/contrib/pgcrypto/pgcrypto.sql.in

[12] M. Bellare, R. Canetti and H. Krawczyk, "Keying hash functions for message authentication", in Proceedings of CRYPTO 1996

[13] http://en.wikipedia.org/Wiki/Mel-frequency_-cepstral_coefficient

Catching Cheats with Interactive Proofs: Privacy-preserving Crowd-sourced Data Collection Without Compromising Integrity

Akshay Dua, Nirupama Bulusu, and Wu-chang Feng
Department of Computer Science
Portland State University
{akshay,nbulusu,wuchang}@cs.pdx.edu

ABSTRACT

Crowd-sourced sensing systems allow people to voluntarily contribute sensor data from mobile devices. They enable numerous applications, including weather and traffic monitoring. However, their proliferation is at risk if the problems of data integrity and privacy persist. People will be reluctant to contribute sensitive information if they cannot trust the system to maintain their privacy, and the system will be reluctant to accept any data transformed to preserve privacy without proof that the transformation was computed accurately. We present an interactive proof protocol that allows an intermediary to convince a data consumer that it is accurately performing a privacy-preserving transformation with inputs from trusted sources, without providing those inputs to the consumer. We provide soundness and correctness proofs for the protocol, discuss its current limitations, and describe its parameters and their effect on data integrity and privacy when tweaked.

1. INTRODUCTION

Integrity of the collected data, and the privacy of data sources are first order concerns for crowd-sourced sensing systems. Such systems can enable critical applications like faster emergency response, and improve health, environment, and traffic monitoring [13, 10, 1]. However, people will be reluctant to volunteer sensitive information (e.g. location, health statistics) if they cannot trust the system to protect their privacy. Conversely, if the volunteered information is first modified to protect privacy, then the system will be reluctant to trust that modification without proof of its integrity.

Consequently, integrity and privacy compete with each other. If the collected data has been previously transformed to preserve privacy (e.g. mixed, aggregated), then a data consumer cannot determine the transformation's integrity unless the raw data used as input is presented as well. However, if the raw data is presented, then the privacy of the data sources gets compromised.

This work attempts to *simultaneously* provide integrity and privacy guarantees for published data without significantly compromising either. The system model assumes a set of trusted data sources that collect and forward sensory data to a *privacy proxy*, which performs a privacy-preserving transformation on the received data, and finally forwards the result to a data consumer. The goal is to assure the data consumer that the proxy indeed computed the expected privacy transformation using data from expected sources (integrity) without providing the consumer with that data (privacy).

Much of the existing work on crowd-sourced sensing, with a focus on integrity and privacy, adheres to this model. Moreover, such a model has the advantage of decoupling the privacy transformation from data collection, enabling transformations that mix data from multiple sources, or perform application-specific data perturbations on the same data. Examples of this model include our earlier work on the design and implementation of a Trusted Sensing Peripheral (TSP) that produces and publishes trustworthy sensory information to a data portal via the user's personal mobile device [5]. The mobile device is allowed to aggregate the data from the TSP before forwarding it. In this case, the mobile device can play the role of the privacy proxy, while the portal plays the role of the data consumer. Other examples include PoolView [6], which introduces the personal privacy firewall to perturb a user's raw data before publishing it to an aggregation service, DietSense [13], which provides private storage to a user where she can edit the images collected from her phone before sharing it further, and AnonySense [11], which uses a trusted server to mix data from at least l clients to provide l-anonymity.

This paper presents an interactive proof protocol [9], using which, only an honest privacy proxy can convince a data consumer that it is correctly computing the expected privacy-preserving transformation, while protecting the privacy of its data sources. The key idea is that unlike traditional interactive proofs with one prover (privacy proxy) and one verifier (data consumer), ours involves a collaboration between the verifier and an additional trusted party (data source) to keep the prover in check. We provide soundness and correctness proofs for the protocol, discuss its current limitations, and describe its parameters and their effect on data integrity and privacy when tweaked.

2. PROBLEM STATEMENT

Only an honest privacy proxy P should be able to convince a data consumer C that it is indeed transforming data $D_j = \{d_{1j}, d_{2j}, ..., d_{nj}\}$, received in interval j, from a set of sources $S = \{s_1, s_2, ..., s_n\}$, using only the transformation function f_{priv}. C receives the result $p_j = f_{priv}(D_j)$, but never the data D_j. The system model is shown in Figure 1.

3. BACKGROUND

Goldwasser et al. [9] introduced the concept of interactive proof systems where a prover P exchanges messages with a verifier V and convinces it with high probability that some

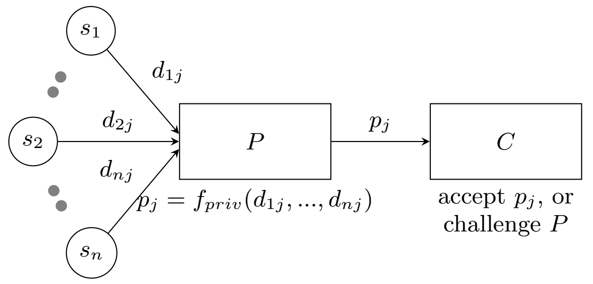

s_1
s_2
s_n
d_{1j}
d_{2j}
d_{nj}
P
p_j
C
$p_j = f_{priv}(d_{1j}, ..., d_{nj})$
accept p_j, or challenge P

Figure 1: System model

statement is true. As a simple example [2], consider how Alice proves to Bob that she can distinguish between Coke and Pepsi. Alice turns her back, then Bob tosses a fair coin, puts Coke or Pepsi in a glass according to the result, and challenges Alice to say which drink it is. Alice tastes the drink and tells Bob which one it is. If Alice *cannot* actually distinguish between the drinks, then she has a chance of being right with probability 1/2. However, when the experiment is repeated k times (each called a "round") her chances are slimmer: $1/2^k$. Thus, when $k = 10$ and Alice answers correctly each time, Bob will be convinced with probability $1 - 1/2^{10}$ or 99.9% that Alice can taste the difference between the drinks. It is key, that the prover (Alice) not know anything about the challenge before it is presented to her by the verifier (Bob). Otherwise, a malicious prover would answer correctly each time. For example, if Alice secretly held a mirror and saw Bob flip the coin and pour out the respective drink, she would simply have to name that drink once Bob challenges her.

Interactive proof systems must be, a) *complete*: any true statement must be accepted by the verifier, and b) *sound*: any false statement must be rejected by the verifier (barring a negligible probability of error). Based on such a proof system, we have constructed a challenge–response protocol that allows a privacy proxy (the prover) to prove to a data consumer (the verifier) that it is honestly performing the privacy transformation, but *without* giving the consumer the inputs to that transformation.

4. THREAT MODEL

Our protocol is designed to prevent a malicious C from learning about the raw data D from the set of trusted sources S, and prevent a malicious P from either using a different transformation: f'_{priv} instead of f_{priv}, or different data: D' instead of D, or both. Since data consumer C has a vested interest in the integrity of the privacy-preserving transformation, it is assumed to be an adversary of privacy, but not of integrity. The privacy proxy P on the other hand is assumed to be an adversary of integrity, but not of privacy. The reason being, that P may actually be controlled by the sources themselves, or maybe trusted by them to not reveal their data. An implication of the above adversarial model is that P and C do not collude in any way.

We do not consider threats from eavesdropping adversaries since standard network security protocols, like TLS, can easily be used to combat them. We also do not consider situations in which P, C, and S do not interact sincerely, implying that neither side suppresses a response expected by the other. Further, we assume S is trusted, its origin is anonymized via a mix network like Tor [4], it can perform anonymous signatures like those described in group signature schemes [3], and that C honestly executes the protocol (since C has a vested interest in collecting high-integrity data). However, C is free to perform offline privacy attacks on the data received from P. Note that our work focuses on *content*, as opposed to, *origin* integrity and privacy.

5. INTERACTIVE PROOF PROTOCOL

As shown in Figure 1, C collects data, transformed using f_{priv} at regular intervals, from sources S, via P. d_{1j} represents the data sent by source s_1 during interval j. Now, in each interval, C can choose to accept the transformed value p_j as-is, or challenge P to prove p_j's integrity. This challenge message marks the beginning of the interactive proof protocol. In Section 5.5, we discuss our protocol's parameters that allow C to challenge all received transformations, or randomly challenge a portion of them.

For simplicity, we explain the protocol using only one trusted data source, say s_1. Consequently, we assume that data d_{1j} is an m-tuple of raw sensory data values $[x_{1j}, ..., x_{mj}]$ collected in interval j. Now, instead of P computing $p_j = f_{priv}(d_{1j}, ..., d_{nj})$ as shown in Figure 1, it will be computing $p_j = f_{priv}(x_{1j}, ..., x_{mj})$. An example of this single-source scenario is a location-based service, where instead of sending raw GPS coordinates d_{1j} from s_1, P sends a coarser region covered by those coordinates. The generalization to multiple sources will be postponed to a more comprehensive version of this paper.

5.1 Constraints

Our protocol must adhere to the following constraints:

1. The data consumer is never provided with the raw data d_{1j}. This is the basic privacy requirement.
2. The data consumer should be able to determine with *high probability* that:
   - The inputs to the privacy transformation f_{priv} consist of only those coming from s_1.
   - No other transformation but f_{priv} is being performed on those inputs.
3. As implied in Section 3, P must not know the challenge *before* it is presented by C. Otherwise, P may tailor its response to pass only that challenge. The key to an interactive proof is that an honest prover can pass both challenges, but a dishonest one can only pass either by guessing.
4. P must not know *beforehand* the interval j in which it will be challenged. Otherwise, it would compute f_{priv} correctly only during those intervals.

5.2 Protocol Details

For the protocol to work, we require a shared symmetric key k_{sc} between the source s_1 and the data consumer C, a buffer at s_1 that is large enough to store data collected in b distinct intervals, and we need to impose the following constraint on the transformation function f_{priv}: given a function $g(r, x)$ that obfuscates input x using random number r, we require that

$$f_{priv}(g(r, x_{1j}), ..., g(r, x_{mj})) = g(r, f_{priv}(x_{1j}, ..., x_{mj})) \quad (1)$$

s_1	**P**	**C**
$j = 1$: *sense* $x_{11}, ..., x_{m1}$		
save $[x_{11}, ..., x_{m1}], j = 1$		
$s_1 \rightarrow P$: $x_{11}, ..., x_{m1}$	$d_{11} = x_{11}, ..., x_{m1}$	
	$P \rightarrow C$: $p_1 = f_{priv}(d_{11})$	p_1
$j = 2$: *sense* $x_{12}, ..., x_{m2}$		
$s_1 \rightarrow P$: $x_{12}, ..., x_{m2}$	$d_{12} = x_{12}, ..., x_{m2}$	
	$P \rightarrow C$: $p_2 = f_{priv}(d_{12})$	p_2
...	...	...

Table 1: Normal Operation

So for example, let $g(r, x) = r \cdot x$, and $f_{priv}(x_{1j}, ..., x_{mj}) = mean(x_{1j}, ..., x_{mj})$, then:

$$\begin{aligned} f_{priv}(g(r, x_{1j}), ..., g(r, x_{mj})) &= mean(r \cdot x_{1j}, ..., r \cdot x_{mj}) \\ &= r \cdot mean(x_{1j}, ..., x_{mj}) \\ &= g(r, f_{priv}(x_{1j}, ..., x_{mj})) \end{aligned}$$

Section 7 discusses the practicality of the above imposed constraint and highlights examples of privacy-preserving transformations that indeed satisfy the constraint. We now discuss the details of the protocol.

Table 1 shows, that while performing its normal sensing duties, source s_1 continues to randomly pick an interval j and save the data $[x_{1j}, ..., x_{mj}]$ collected in that interval. Once the buffer is full (has b intervals worth of data) s_1 uniformly and randomly selects an interval of data from its buffer (Table 2). It then sends one of two sets of messages to P and C corresponding to each challenge, after which a simple indicator challenge message from C to P begins the interactive proof. Once the proof is complete, s_1 can purge the respective interval of data from its buffer, thus making room for data from the next randomly picked interval. Other notation includes $E_{k_{sc}}$, which is some symmetric encryption scheme using key k_{sc}.

The proof starts when C challenges P about the integrity of some transformed result p_j received in the past (Table 2); we call this p_j a commitment from P. However, which interval j is challenged, is decided by s_1 and selectively revealed to C. The goal is to not let C have j and the obfuscated values M_2 together. Otherwise, it can recover r and therefore the raw values from s_1. What is noteworthy, is that unlike traditional interactive proofs with one prover and one verifier, ours involves a collaboration between the verifier C and an additional trusted party (the data source) to keep the prover P in check.

There are two types of challenges and associated tests that take place. Message M_1 corresponds to the first type of challenge, while M_2 to the second type. The first type is designed to test the integrity of P's past commitment, while the other tests the computation of f_{priv} itself. The padding PAD assures that M_1 and M_2 have the same length. The encryption further ensures that the messages are statistically similar. That way, a malicious P intercepting message M_1 or M_2 will be unable to discern them and hence, not know which type of test will be performed by C.

Intuitively, the protocol is trying to ensure that C does not have r or j, and the obfuscated message M_2 together in any given interval j. Otherwise, C could first recover r using the property in Equation (1), and then recover s_1's raw data values from M_2 (remember that $g(r, x)$ only obfuscates x, and is not a cryptographic one-way function).

At the same time, the protocol wants to ensure that P does not know what type of test is going to be performed by C. Otherwise, P might tailor its response to pass only that test. For example, assume that in each interval, a malicious P fabricates the transformation it sends to C. Also assume, that it caches all raw data values ever received from s_1, and computes and caches (but does not send to C) the correct p_j for each interval as well. Now, suppose it knows that challenge 1 is underway. Then, it computes p as required by the challenge, recovers some r (not necessarily the right one) using $p = g(r, p_j)$ for some interval j, extracts each raw value from the obfuscated message M_0, and compares the set of raw values with those cached for that interval. If a match is found, P knows it has the correct j and therefore r, otherwise it picks another interval j and repeats this process till a match is found. With the correct value of r, it can provide C with the expected response p and pass challenge 1 each time. Now, if P did not know which challenge was underway, then the above attack (called the *find-r* attack) can only pass challenge 1 but not challenge 2. So, with probability 1/2 a malicious P's response to the challenge would be rejected by C. However, an honest P's response could simultaneously pass both tests.

5.3 Analysis

Without going into details, we now point out that our protocol satisfies all constraints mentioned in Section 5.1, but the second one. The second constraint can be satisfied by parametrizing the protocol, and this is discussed in the next section. We now show that our interactive proof is *complete* and *sound*.

Completeness is easier to prove. From Table 2, we can see that any honest P that computes and publishes $f_{priv}(d_{1j})$, and then $f_{priv}(g(r, x_{1j}), ..., g(r, x_{mj}))$ when challenged, will indeed pass either of C's integrity tests with its response p.

We prove soundness by case analysis. But first, we can safely assume that $g(r, x)$ is honestly computed. Since during either challenge, one honest party, C (honest with respect to executing the protocol) or s_1 (trusted), computes g. Thus, if P computes some g' instead of g it will fail either challenge. There are three cases we must consider: P may use fabricated inputs to the transformation f_{priv}, may use the wrong transformation, or both. Throughout, we will refer to notation used in Table 2.

Case 1. Fabricated inputs: If a malicious P publishes $p_j = f_{priv}(d'_{1j})$ instead of $p_j = f_{priv}(d_{1j})$, then during the challenge it must still prove to C that $p = g(r, p_j)$. If it honestly computes p, then this check will fail. If P mounts an attack similar to *find-r* described in Section 5.2, it can pass C's first test, but not the second. Since in the second test, M_2 comes directly from the trusted source s_1.

Case 2. Wrong transformation: If a malicious P is computing $f'_{priv}(d_{1j})$ instead of $f_{priv}(d_{1j})$, then to pass either test, f'_{priv} must share the same relationship with g that f_{priv} does (Equation (1)). Now say that it does, then P could pass C's first test, but again, not the second one. Since in the second test, C itself computes f_{priv}. Also, P's *find-r* attack suffers the same fate.

Case 3. Wrong inputs and transformation: If a malicious P is computing $f'_{priv}(d'_{1j})$ instead of $f_{priv}(d_{1j})$, then the *find-r* attack could pass test 1, but not test 2. Since in test 2, C computes f_{priv} with trusted inputs from

s_1	**P**	**C**
From b saved intervals, randomly *pick* j Then, *pick* random number r Then, with probability 1/2, *either* send: $s_1 \rightarrow C$: $M_1 = E_{k_{sc}}(r, j, PAD)$ $s_1 \rightarrow P$: $M_0 = g(r, x_{1j}), ..., g(r, x_{mj})$ OR send: $s_1 \rightarrow C$: $M_2 = E_{k_{sc}}(g(r, x_{1j}), ..., g(r, x_{mj}))$ $s_1 \rightarrow P$: $M_0 = g(r, x_{1j}), ..., g(r, x_{mj})$		
	M_0 On receiving challenge, $p = f_{priv}(M)$ $P \rightarrow C$: p	M_1 OR M_2 $\leftarrow$ Challenge P p

		Test 1 (if received M_1):
		$r, j, PAD = D_{k_{sc}}(M_1)$
		if $p \neq g(r, p_j)$, **reject**
		
		Test 2 (if received M_2):
		if $p \neq f_{priv}(D_{k_{sc}}(M_2))$, **reject**

Table 2: Interactive proof of integrity for privacy-preserving transformations performed by P

source s_1.

5.4 An Attack on Privacy

It is possible for C to launch an offline privacy attack on the $g(r, x_{1j}), ..., g(r, x_{mj})$ data values it receives during challenge 2. As mentioned before, g is only assumed to be an obfuscation function that can easily be reversed. Leading to ways in which r could be retrieved, and then possibly the raw data values x_{ij}, $1 \leq i \leq m$.

This attack is the same as *find-r* described earlier, but without the cache of raw trusted data values from the source. Without this cache, C will not know if the retrieved raw values are actually correct. It would have to *guess* if the retrieved raw data values seems "plausible". This attack can be mitigated by increasing the number of possible choices from which to guess. We discuss this further in Section 5.5. In the worst case, C *could* obtain raw data values for 1/2 the number of intervals in which it challenges P. Furthermore, the plain-text guessing required in this attack may require human intervention, making this attack more costly.

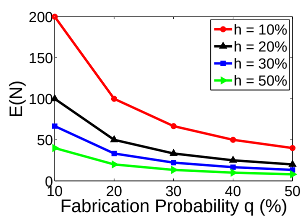
5.5 Parameters

Table 2 describes only one round of the interactive proof. Using multiple rounds, C can gain more confidence in the integrity of the published value p_j being challenged. Further, by challenging more often, C can gain more confidence in the integrity of the data stream in general. Based on such observations, we have defined the following parameters:

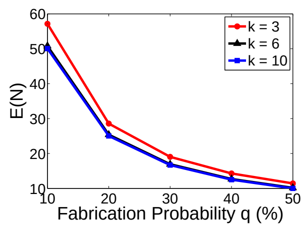
- h: the percentage of intervals that C will challenge P. So, if h is 20%, then C will randomly challenge P during one of every five intervals. Note that s_1 must know h as well, since it initiates the challenge. A larger h, will provide more integrity, but will reduce privacy since C will receive more obfuscated values that it can potentially attack offline.

- k: the number of rounds each interactive proof is executed. As mentioned in Section 3, with $k = 10$, C can have 99.9% confidence in the integrity of the published value being challenged. Closely related to k, is an interesting metric we call the *confidence index* of P: defined as the number of challenges P has passed, over the total times challenged. When k is low (say 1, i.e. 50% chance a published value is incorrect), and P's confidence index is high (say 100%) then C can still be sure about the integrity of the already published values from P. However, if P's confidence index is low, then C can choose to increase k and at least gain more confidence each time it challenges P.

- b: the number of intervals of data that s_1 needs to save before initiating challenges. Increasing b can mitigate the privacy attack discussed in the previous section. It will increase a malicious C's ambiguity about the correct inputs to f_{priv} that might have created a given p_j. For a given interval j, C would have to choose between b/h possible sets of inputs. For example, if $b = 1000$ and $h = 1/5$, then C would have to guess the right inputs from 5000 possible ones. In the worst case, C can retrieve $h/2$ intervals of raw data.



(a) Varying h, $k = 1$.



(b) Varying k, $h = 20\%$.

Using parameters h and k we have the following equation for the expected number of intervals $E(N)$ before C detects a malicious P. Here, P is fabricating a transformed value p_j with probability q.

$$E(N) = \sum_{n=1}^{\infty} n \times (1 - hq(0.5^k))^{n-1} \times hq(0.5^k) \quad (2)$$

Plots of $E(N)$ while varying either h or k are shown in Figures 2(a),2(b). When $h = 20\%$, $q = 10\%$ a malicious P is expected to fail a challenge in 100 intervals, and publish 10 fabricated transformed values. Also, by setting $k = 3$ (instead of $k = 1$) we can reduce time to detection by $\approx 40\%$.

6. RELATED WORK

Much of the previous security oriented work in crowd-sourced sensing has focused *either* on data integrity, or privacy. PoolView [6] enables community statistics to be computed using perturbed private data, but trusts its users to honestly perturb that data. PriSense [14] employs data slicing and mixing to provide privacy while still supporting a wide variety of aggregation functions. However, it is assumed the functions themselves are honestly computed. Our previous work on Trusted Sensing Peripherals [5] supports high-integrity aggregation, but does not provide privacy guarantees.

VPriv [12] makes a strong attempt to offer integrity and privacy for computing tolls over paths driven by vehicles. However, due to the use of an additive homomorphic commitment scheme, VPriv can only guarantee the integrity of additive functions. Additionally, the random spot checks needed to keep drivers honest may compromise privacy.

Fully homomorphic encryption schemes (addition and multiplication operations supported over ciphertexts) could go a long way in solving the integrity and privacy problem. The first such scheme was recently introduced by Gentry et al. [7]. However, computation on ciphertexts is still widely considered to be computationally expensive.

7. LIMITATIONS

It remains to be seen what types of privacy-preserving transformations can work with our protocol, given the constraint in Equation (2). We have shown that a transformation computing the mean of its inputs can be used. Other transformations we plan to investigate include mixing data from multiple sources to provide k-anonymity [15], and possibly location blurring.

Interactive proofs require their participants to be online. Hence, the data sources need to be online while the proof is taking place. Future work could include constructing a non-interactive proof that achieves the same goals.

Interactive proofs can be zero-knowledge [8] if no other information but the truth of the statement being proved is revealed. Unfortunately, our protocol falls short of this goal because data privacy could *possibly* (not surely) be compromised by a malicious data consumer (see Section 5.4).

8. CONCLUSION

Crowd-sourced sensing has a bright future, but both the integrity of the collected data, and the privacy of data sources are always at risk. Without integrity assurances, data consumers like the government or researchers will be reluctant to use the data, and without privacy assurances, people will be reluctant to contribute the data.

We have proposed a possible solution using interactive proofs that *simultaneously* addresses the conflicting problems of integrity and privacy. The interactive proof allows an intermediary to convince a data consumer that it is accurately performing a privacy-preserving transformation with inputs from trusted sources, without providing those inputs to the consumer. Sources can be trusted when, for example, they provide verifiable attestations to the integrity of sensed data with the aid of integrated trusted platform modules [5]. The key idea is that unlike traditional interactive proofs with one prover (privacy proxy) and one verifier (data consumer), ours involves a collaboration between the verifier and an additional trusted party (data source) to keep the prover in check. We have provided soundness and correctness proofs for the protocol, discussed its limitations, and described its parameters and their effect on data integrity and privacy.

9. ACKNOWLEDGMENTS

This work was supported by the National Science Foundation under grants CISE-0747442 and CNS-1017034. We would like to thank Tom Shrimpton for his valuable feedback on the contents of this paper.

10. REFERENCES

[1] E. Agapie, E. Howard, J. Ryder, A. Steiner, D. Lam, R. Rosario, A. Modschein, D. Houston, J. Burke, M. Hansen, et al. PEIR. 2007.

[2] B. Barak. Lecture 15: Zero Knowledge Proofs. `www.cs.princeton.edu/courses/archive/spring10/cos433/lec17new.pdf`, Nov 2007.

[3] D. Chaum and E. Van Heyst. Group Signatures. *Berlin: Springer-Verlag*, 265, 1991.

[4] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *USENIX Security*, pages 21–21. USENIX Association, Berkeley, CA, USA, 2004.

[5] A. Dua, N. Bulusu, W. Feng, and W. Hu. Towards Trustworthy Participatory Sensing. In *HotSec'09: Proceedings of the 4th USENIX Workshop on Hot Topics in Security*. USENIX Association Berkeley, CA, USA, 2009.

[6] R. Ganti, N. Pham, Y. Tsai, and T. Abdelzaher. PoolView: stream privacy for grassroots participatory sensing. In *Proceedings of ACM SenSys*, pages 281–294, Raleigh, North Carolina, 2008. ACM.

[7] C. Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st annual ACM symposium on Theory of computing*, pages 169–178. ACM, 2009.

[8] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM (JACM)*, 38(3):728, 1991.

[9] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems. In *Proceedings of the seventeenth annual ACM symposium on Theory of computing*, page 304. ACM, 1985.

[10] B. Hull, V. Bychkovsky, Y. Zhang, K. Chen, M. Goraczko, A. Miu, E. Shih, H. Balakrishnan, and S. Madden. Cartel: a distributed mobile sensor computing system. In *Proceedings of ACM SenSys*, pages 125–138, Boulder, Colorado, 2006. ACM.

[11] A. Kapadia, N. Triandopoulos, C. Cornelius, D. Peebles, and D. Kotz. AnonySense: Opportunistic and Privacy Preserving Context Collection. *LNCS*, 5013:280, 2008.

[12] R. Popa, H. Balakrishnan, and A. Blumberg. VPriv: Protecting privacy in location-based vehicular services. In *Proceedings of the 18th Usenix Security Symposium*, 2009.

[13] S. Reddy, A. Parker, J. Hyman, J. Burke, D. Estrin, and M. Hansen. Image browsing, processing, and clustering for participatory sensing: lessons from a DietSense prototype. In *ACM SenSys*, pages 13–17, Cork, Ireland, 2007. ACM.

[14] J. Shi, R. Zhang, Y. Liu, and Y. Zhang. PriSense: Privacy-Preserving Data Aggregation in People-Centric Urban Sensing Systems. In *IEEE INFOCOM*, 2010.

[15] L. Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty Fuzziness and Knowledge Based Systems*, 10(5):557–570, 2002.

Opportunistic privacy preserving monitoring*

Luca Becchetti
Sapienza University of Rome
Via Ariosto 25
Rome, Italy
becchetti@dis.uniroma1.it

Luca Filipponi
Sapienza University of Rome
Via Ariosto 25
Rome, Italy
filipponi@dis.uniroma1.it

Andrea Vitaletti
Sapienza University of Rome
Via Ariosto 25
Rome, Italy
vitale@dis.uniroma1.it

ABSTRACT

In this paper we present an approach to support privacy in opportunistic sensing. In particular, we use compact and privacy preserving representations of data (or sketches), that allow us to compute relevant statistics over data without disclosing users' sensitive information (e.g. locations). We exemplify our approach by referring to the important application of noise pollution monitoring. We present preliminary experimental results showing that sketches can actually be employed to produce accurate environmental maps, at the same time preserving users' privacy.

1. INTRODUCTION

Opportunistic people-centric sensing [12] has been gaining popularity, with several systems and applications being proposed to leverage users' mobile devices to collectively measure environmental data. In such systems, nodes report sensor data through opportunistic network connections, such as third-party access points. However, people centric sensing generally suffers from privacy related issues, namely the need to share data provided by users without disclosing any sensitive information about user's privacy (e.g., locations).

In this paper we present an approach to support privacy in opportunistic sensing. In particular we use sketches, namely compact and privacy preserving representations of data, that allow us to compute relevant statistics over the data without disclosing users' sensitive information. The proposed techniques can be exploited in several application domains such as environmental monitoring, analysis of social patterns, traffic maps etc. Here we exemplify our approach referring to the relevant application of noise pollution monitoring.

The Directive 2002/49/EC of the European Parliament has made the avoidance, prevention, and reduction of environmental noise a primary issue in European policy and the Commission required Member States to provide accurate noise pollution maps. Today's noise measurements are mainly carried out by designated officers that collect data in a location of interest. Even if this assessment procedure is still compliant with European regulations [9], it often fails to provide scalable and accurate estimations of the real noise pollution levels. Nowadays, the applicability of fixed wireless sensor networks [10] to wide area long-term monitoring, is still limited due to its high installation and maintenance costs.

*Partially supported by EU STREP Projects ICT-215270 FRONTS, ICT-257245 VITRO and by PRIN 2008 research project COGENT.

2. RELATED WORK

People-centric sensing [7][8] has leveraged the use of human carried devices (such as smart-phones) to sense information directly or indirectly related to human activity or environment, in an opportunistic or participatory way. The MetroSense project [14] is working with industry and agencies to develop new applications, classification techniques, privacy approaches, and sensing paradigms for mobile phones enabling a global mobile sensor network capable of societal-scale sensing. Most related to our reference application, the NoiseTube [13] project has developed a novel platform for the monitoring of urban noise pollution, based on mobile phones. The same approach has been followed by the NoiseSpy [11] project in which noise maps are built on the basis of data coming from users' devices in a participatory way. Both these projects require users to agree and share informations regarding noise levels measurements, together with their position in order to allow geotagging to an external system. While both these works demonstrate the feasibility of the use of smartphones/cellphones as sound meters, their platforms suffer of a major lack of privacy for involved users, thus allowing an attacker to trace users' movements. Privacy preservation in location based services has already been addressed by [16, 15]. In [15], accurate traffic speed maps in a small campus town are build from shared GPS data of participating vehicles, where the individual vehicles are allowed to "lie" about their actual location and speed at all times. In our approach, data are always correct but represented in a compact and privacy preserving way (i.e sketches). Differently from [16], where data are available in clear to the intended receiver, in our work sketches allow a central authority to select relevant traces to reconstruct an accurate map, but without revealing to anybody (central authority included) relevant information on users' positions.

3. SYSTEM OVERVIEW

Taking inspiration from the NoiseTube [13] set-up, we aim to leverage user's smartphones to sample the environment and provide collected data to a central authority. Nevertheless contrary to NoiseTube, our solution is based on *opportunistic monitoring* and explicitly considers *privacy preservation* a primary concern to actually encourage users' participation. Nowadays mobile phones are personal devices, primarily intended to serve the user with telephony, messaging and other functionalities. Additional services like noise monitor-

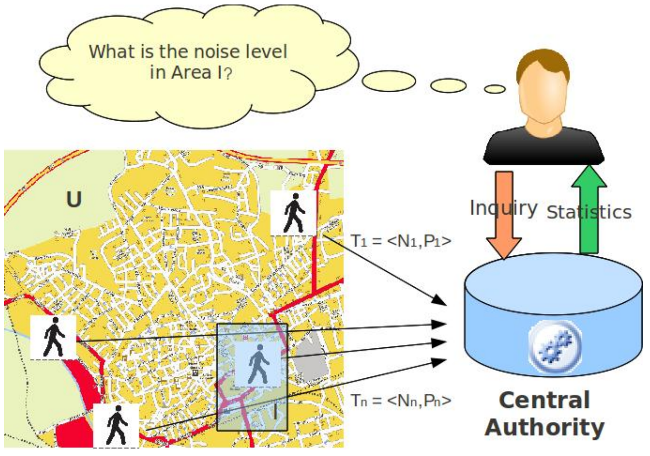


Figure 1: System Overview

ing cannot be considered of primary importance, and thus people would likely not support them if they will negatively affect the behavior of the "primary" services. For this reason, we think a mobile phone application should be transparent and communicate opportunistically whenever users get connected for their own purposes. This approach strongly limits the availability of real-time data, because data are only transmitted whenever a connection is available. However, for the statistical purposes of the noise monitoring application we are discussing here, this is not a major problem. Instead, we do believe that a mandatory requirement to achieve a significant users' participation is the preservation of their privacy. Sampled data must be geo-referenced to be of some utility, as a consequence users' movements could be easily traced with a serious loss of privacy. Thus, a privacy preserving representation of data is needed, but at the same time, operations on sampled data should be still possible to the central authority, in order to select relevant samples among the whole amount of received data for each inquiry. The noise monitoring service we envision, foresees the participation of three main actors: mobile users, central authority and system users. **Mobile users** are responsible for noise data collection. They participate to the service by running a noise monitoring application on their phones (see section 3.1). This application exploits the on board microphone to sample environmental data while the phone is idle, so as not to affect phone's normal usage. Whenever the user establishes a connection, environmental data are opportunistically sent to the **central authority** which is in charge to elaborate statistics and to answer queries issued by **system users** on the average noise level in an area of interest.

More in detail, we consider n users moving in an area U and collecting traces on environmental noise. The trace generated by user i, denoted by T_i consists of a set of pairs $< n_i^t, p_i^t >$ where n_i^t and p_i^t are respectively the observed noise level and the location of the user in U at time t. Let $P_i = \{p_i^t\}, \forall t$ and $N_i = \{n_i^t\}, \forall t$, be respectively the set of positions (i's *position set*) occupied by the user over time and the corresponding set of noise levels (i's *noise set*). We stress that, in order to guarantee users' privacy, P_i should not be disclosed to third parties, including the central authority, and thus it should be represented in a suitable privacy preserving format. Instead, the set of noise values must be publicly available, since it is used for the estimation of the average noise level. To achieve this, user i sends to the central authority the pair $(\mathbf{Sk}(P_i), N_i)$, where $\mathbf{Sk}(P_i)$ is a *sketch* (i.e., a suitably generated compact summary) of P_i. In Subsection 3.2 we show how to generate $\mathbf{Sk}(P_i)$ so that it has the following properties: i) $\mathbf{Sk}(P_i)$ represents P_i implicitly in small space (in the order of 10^2 bytes at most) and does not allow to infer P_i; ii) considered any area I of interest, $\mathbf{Sk}(P_i)$ allows the central authority to estimate the extent to which the set P_i of i's positions covers I. Note that this is achieved using only $\mathbf{Sk}(P_i)$, so that P_i is never explicitly disclosed to third parties. All users' traces $\mathcal{T} = T_1, T_2, .., T_n$ are made available to the central authority according to the mechanism described above. This, in turn collects the traces and performs elaboration to answer queries issued by system users concerning the average noise level in any area $I \subseteq U$ of interest. Let's denote by $\mathcal{Q}(I)$ the query asking for the average noise in area I. Upon reception of $\mathcal{Q}(I)$, the central authority selects the minimum number of P_i's guaranteeing coverage of I and then calculates the average noise level over time in that area. This is the classic set cover problem , but contrary to classic set cover we have to enforce some kind of privacy preserving technique to encourage users participation to the monitoring activity. In order to select the minimal subset of user traces that cover I, the central authority uses the techniques described in Section 4. In performing this computation, it only uses the sketches $\mathbf{Sk}(P_i)$ of users' position sets and not the position sets themselves.

3.1 Mobile phones as sound meters

The feasibility of using Mobile phones as sound meters has already been discussed in [13, 11]. Both these works, presented a mobile phone application that, requiring user participation, logs sound pressure values using the onboard microphone and some correction algorithms, thus obtaining a limited estimation error. Shifting from participatory to opportunistic monitoring, additional challenges have to be discussed. In opportunistic contexts indeed, sampled values can be affected by errors generated by external and unpredictable noise sources. As an example, consider a user carrying the mobile phone in her pocket: noise values could suffer from attenuation effects or spikes due to the noise generated by objects in the pocket (e.g., coins, keychains, voices). In such a scenario, an additional filtering technique is needed in order to obtain more accurate data. We defer the study of these aspects to future work, and here we focus on the evaluation of our system from an algorithmic point of view.

3.2 Privacy preserving data representation

As discussed before, in the application we envision, a user only sends a compact summary of her position set, from which it is hard to recover the original set. In this section we present a class of sketches [6, 5, 4] that, while compact and addressing the privacy issues mentioned above, allow the (approximate) implementation of some basic primitives on sets (such as union and intersection) that are required to implement the algorithms presented in section 4. In the rest of this subsection we present techniques used by mobile users' terminals to produce compact summaries of their respective position sets.

Compact representation of sets: We only briefly outline the principles underlying the technique we propose, leaving out many theoretical aspects for the sake of brevity. The interested reader can refer to [6, 5, 4]. In the remainder of this subsection, we consider without loss of generality subsets of $[n] = \{0, \ldots, n-1\}$, for a suitable integer n. We briefly note that standard techniques allow us to reduce to this situation in all practical cases[1].

Assume we have a family $\mathcal{H}$ of hash functions such that: i) every $h \in \mathcal{H}$ produces a permutation of $[n]$; ii) if h is chosen uniformly at random from $\mathcal{H}$ the following holds: for every set $S \subseteq [n]$:

$$\mathbf{P}[x = \arg\min(\pi(S))] = 1/|S|, \forall x \in S.$$

Such a family is said *minwise independent* [4]. In practice, minwise independent hash functions are hard to generate, since they require a high number of truly random bits. In this paper, we use functions of the form $h(x) = ((ax + b) \mod c) \mod n$ [3], that excellently approximate minwise independent families. Here, c is a large prime (e.g., the well-known Mersenne prime $2^{32} - 1$) and n is the number of possible locations in U. Finally, $a \in \{1, \ldots, c-1\}$ and $b \in \{0, \ldots, c-1\}$.

Sketch generation and maintenance: Considered any subset S of $[n]$, we construct its sketch as follows: for m times, we choose, independently, uniformly and at random, a hash function from a minwise independent family. Let $H_i(x)$ the i-th function chosen and let $\min_i(S) = \min_{x \in S} H_i(x)$. Then $\mathbf{Sk}(S) = \{\min_i(S), \ldots, \min_m(S)\}$. In our case, we consider hash functions of the form $h(x) = ((ax+b) \mod c) \mod n$. In practice, generating such a hash function means generating a and b uniformly at random from $\{1, \ldots, c-1\}$ and $\{0, \ldots, c-1\}$ respectively.

Sketch properties: Given sets S_1 and S_2, the sketch of $S_1 \cup S_2$ can be immediately obtained from $\mathbf{Sk}(S_1)$ and $\mathbf{Sk}(S_2)$ as follows: $\mathbf{Sk}(S_1 \cup S_2) = \{M_1, \ldots, M_m\}$, where $M_i = \min\{\min_i(X_1), \min_i(X_2)\}$. Another interesting property of these sketches is that they allow to easily and accurately estimate the Jaccard coefficient of two sets, a standard measure of the similarity between sets, widely used in information retrieval. Given two subsets S_1 and S_2 of $[n]$, their Jaccard coefficient is defined as $J(L(S_1), L(S_2)) = \frac{|L(S_1) \cap L(S_2)|}{|L(S_1) \cup L(S_2)|}$. It can be shown [4] that for every S_1, $S_2 \subseteq [n]$: $\mathbf{P}[\min(\pi(S_1)) = \min(\pi(S_2))] = J(S_1, S_2)$. This suggests a simple statistical estimator of the Jaccard coefficient of two sets, which we discuss in the next paragraph. To estimate $J(S_1, S_2)$, we simply consider their sketches $\mathbf{Sk}(S_1)$ and $\mathbf{Sk}(S_2)$ and let $C_m = |\{i : \min_i(S_1) = \min_i(S_2)\}|$. Then, a simple probability argument allows to show that C_m/m is an increasingly accurate estimate of $J(S_1, S_2)$.

Compact representation of position sets: All mobile users will use the same set $H_1(\cdot), \ldots, H_m(\cdot)$ of minwise independent hash functions. These will be generated by the central authority and then sent to each mobile user once, i.e., the first time she joins the application. Note also that, in practice, the linear functions we use are represented in terms of a small set of parameters. For example, if we use 100 hash functions, each mobile user will need to receive 202 integer values (the coefficients a and b of each hash function plus c and n), for a total of less than 1 KByte, if we represent integers using 4 bytes. Then, mobile user i will generate sketches of her position sets as follows: her sketch $\mathbf{Sk}(P_i)$ is initially set to $\{0, \ldots, 0\}$. Let $\{M_1, \ldots, M_m\}$ be i's sketch at some point. If she moves to a new position p (e.g., identified by the GPS coordinates of a new base station she connects to), then $\mathbf{Sk}(P_i)$ is updated as follows: $M_j = \min\{M_j, h_j(p)\}, \forall j = 1, \ldots, m$. This sketch update corresponds to updating i's position set as follows: $P_i = P_i \cup \{p\}$. This representation of position sets would require an attacker willing to recover P_i knowing $\mathbf{Sk}(P_i)$ to generate a sketch for all the possible P_i in the world U (even with varying size) and estimate the Jaccard coefficient between them. The more is the size of U, the more an attack is unfeasible.

4. PROBLEM STATEMENT

The problem of finding the minimum number of traces covering the area I of interest for a system user, can be formulated as an instance of the $NP-complete$ set cover problem. In the classical set cover problem we are given a set I, taken from a universe U, and a collection $\mathcal{T} = T_1, T_2, .., T_n$ of subset of U. The pair $(U, \mathcal{T})$ is sometimes called a set system. The aim is to compute a sub-collection $\mathcal{T}' \subseteq \mathcal{T}$ which covers I with minimum cost, namely using the smallest number of sets in $\mathcal{T}$.

In this section we first recall the Greedy Algorithm for set cover and then, after providing a few useful remarks, we introduce our algorithm that implements the greedy set cover using sketches.

4.1 Greedy Algorithm for Set Cover with Unitary Costs

This algorithm is given in figure 2[2].

```
Algorithm Standard-Greedy
Require: set system $(\mathcal{T}, U)$
1: $C = \emptyset$ ($C$ contains identifiers of sets in set cover)
2: $\hat{\mathcal{T}} = \mathcal{T}$
3: $\hat{S} = \arg\max_{S \in \hat{\mathcal{T}}} |S \cap (U - C)|$
4: while $|S \cap (U - C)| > 0$ do
5:     $\hat{\mathcal{T}} = \hat{\mathcal{T}} - \{\hat{S}\}$
6:     $C = C \cup \hat{S}$
7:     $\hat{S} = \arg\max_{S \in \hat{\mathcal{T}}} |S \cap (U - C)|$
8: end while
9: return $C$
```

Figure 2: Greedy Algorithm for Set Cover.

Since it seems hard to give the sketch of the difference of two

[1]In our case, the position set P_i of a user i is a finite set of geographical positions (e.g., GPS coordinates). As such, it can be put in correspondence with a subset of the integers using standard techniques. E.g., [4] shows how to achieve this for Web documents using Rabin's fingerprinting method.

[2]In order to make the pseudo-code more readable, we slightly abuse notation, since we regard C as both a set of sets (the set cover) in line 9 and as the union of the sets that form the cover in lines 3, 4, 6 and 7. Analogous considerations hold for Algorithm 3.

sets given the sketches of the two sets, we slightly modify the algorithm above, by replacing $|S \cap (U - C)|$ with $|(S \cup C) \cap U|$ in steps 3, 4 and 7 of Algorithm 2. Maximizing the former quantity is equivalent to maximizing the latter.The proof is trivial, and is omitted due to space constraints.

4.2 Greedy Set Cover Algorithm using sketches.

We next describe the algorithm PP-Greedy. This algorithm is an implementation of the standard Greedy Set Cover algorithm for the case of unitary set costs. The novelty is that it is "rephrased" in terms of operations on set sketches instead of the sets themselves. Algorithm 3 is the sketch-based counterpart of Algorithm 2.

Essentially, in lines 5 and 11, instead of considering the maximization of $|(S \cup C) \cap U|$, we choose the set S, such that the (estimated) Jaccard coefficient between $S \cup C$ and U is maximized. This, up to approximations, is the set S such that $\mathbf{Sk}(S \cup C)$ and $\mathbf{Sk}(U)$ share the largest number of equal minima, i.e., the set that maximizes $Eq(U, C \cup S)$, where $Eq(U, C \cup S) = \sum_{i=1}^{m}(\min_i(U) == \min_i(C \cup S))$, namely the number of times the minima of U and $C \cup S$ agree.

Algorithm PP-Greedy

```
Require: Sketch $\mathbf{Sk}(S_i)$, for $i = 1, \ldots, |\mathcal{T}|$, $\mathbf{Sk}(U)$
 1: $E = 0$
 2: $C = \emptyset$ ($C$ contains identifiers of sets in set cover)
 3: $\mathbf{Sk}(C) = \{\infty\}_{i=1,\ldots,m}$
 4: $\hat{\mathcal{T}} = \mathcal{T}$
 5: $\hat{S} = \arg\max_{S \in \hat{\mathcal{T}}} Eq(U, C \cup S)$
 6: $\hat{E} = Eq(U, C \cup \hat{S})$
 7: while $\hat{E} > E$ do
 8:     $E = \hat{E}$
 9:     $\hat{\mathcal{T}} = \hat{\mathcal{T}} - \{\hat{S}\}$
10:     $C = C \cup \hat{S}$
11:     $\hat{S} = \arg\max_{S \in \hat{\mathcal{T}}} Eq(U, C \cup S)$
12: end while
13: return $C$
```

Figure 3: Privacy Preserving Greedy Algorithm for Set Cover.

5. PRELIMINARY RESULTS

In this section we discuss the results of a preliminary experimental activity performed to verify the performance of PP-Greedy vs Greedy on synthetic traces.

Generating traces: Our experiments are based on mobility traces generated through the Global Mobility Simulation Framework (GMSF) [1] developed at ETH Zurich. It allows to generate mobility traces according to different models, such as Random Waypoint, Manhattan, or GIS Based. In our experiments, 10, 50 and 100 mobile users move in a square U of side length $d = 1000$ units, according to the Manhattan model; each user generates a trace made of a total of 35000 positions . We adopted the Manhattan model, because it is the most suited to describe the mobility patterns of users in a urban scenario. Moreover, in order to make the simulation more realistic for an opportunistic scenario, whenever a user stops in the simulated environment, it generates a sketch of the set of positions P_i she went through so far, and opportunistically sends it to the central authority. In other words, each user sends to the central authority a set of sketches $Sk(P_i)$; each of them representing a subset of the 35000 positions sampled by the user.Sketches are generated starting from the set of positions P_i applying the technique described in section 3.2 with $m = 100$ hash functions; this number of hash functions provides the best trade-off between accuracy and size of the sketches [2]. The resulting sketch is an array of 100 integers, with a total size of 400 bytes.

The area of interest I considered for the experiments are the crossroads in the Manhattan topology, each one centered on the diagonal of U (from the center to the top-left corner of the square) made of 200 positions.

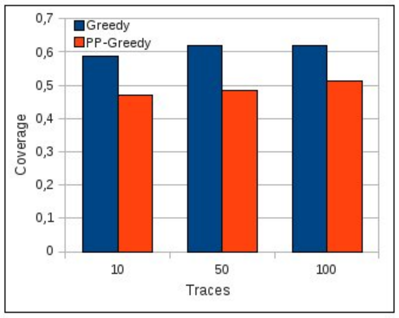
Metrics: The Greedy algorithm Gr receives in input the area of interest I and the set of sets of positions sampled by the users $P = P_1, .., P_n$, and provides as output the set $P_{Gr} \subseteq P$ that approximate the set cover of I. The PP-Greedy algorithm $PP - Gr$, instead of the set P, receives in input $Sk_P = (Sk(P_1), .., Sk(P_n))$, and provides as output the set $P_{PP-Gr} \subseteq Sk_P$ that approximate the set cover of I. We evaluate the performance of Gr and $PP - Gr$ considering three metrics. The *cardinality* of output i.e., the number of position sets used to cover the area of interest, the *coverage* of the output, intended as the fraction of positions in the area of interest that are covered by the output.[3]. The *error*, defined as the fraction of positions in the output which are not in I. As an example consider the following sets $I = 1, 3$, $P_{Gr} = 1, 2, 3, 4$. In this case the cardinality is 2, the coverage is 100% and the error is 50%.

Results: As expected and as figure 4(a) shows, the coverage achieved by both algorithms slightly increases with the number of participating mobile users (i.e. number of traces). The behaviors of both algorithms are similar, but PP-Greedy always achieves 10% less coverage than Greedy; this is the consequence of the loss of information due to the use of sketches instead of the explicit position sets. More surprisingly, the cardinality of PP-Greedy outputs is remarkably lower, approximately half the cardinality of Greedy (see figure 4(b)). The higher cardinality of the Greedy solution results in an increased error of Greedy, as can be observed in figure 4(c). This is due to the fact that each new set added to the solution contributes with a minimum number of positions . Thus, increasing the cardinality of the solution in general improves the coverage, but when the coverage is already high, each new set added to the solution will have an increasingly higher chance of contributing with new positions that do not belong to the area of interest, thus increasing error. This explains why the PP-Greedy's error is always lower than 15%, while Greedy's error is always higher than 25%. There seems to exist a "breaking-point" for the solution, beyond which the addition of more P_i's to the solution slightly increases coverage, but at the same time it significantly increases error.

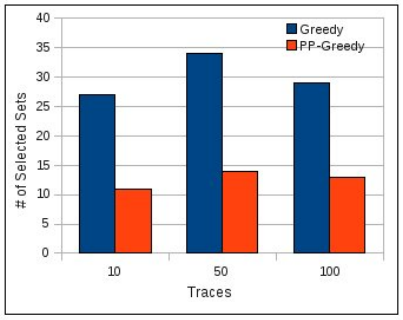
6. CONCLUSIONS AND FUTURE WORK

The lower accuracy of PP-Greedy is fairly compensated by the lower error and cardinality of its outputs; as suggested

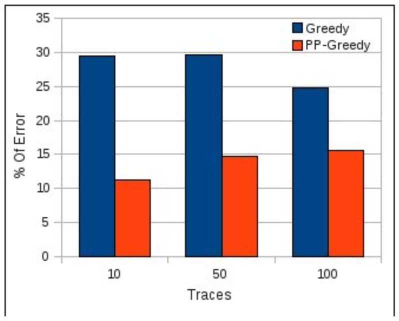
[3]We calculate the coverage of P_{PP-Gr} considering the set of corresponding positions



(a) Coverage



(b) Cardinality



(c) Error

Figure 4: For every area of interest and for every performance metric, we averaged the results over 10 **runs of both algorithms.**

by our results, there is an interesting trade-off between accuracy on one-side and cardinality and error on the other. This observation seems to support the conclusion that PP-Greedy algorithm is a good privacy preserving approximation of Greedy but at the same time this preliminary results deserve future investigations about the effects of tuning the granularity of P_i's with respect to the dimension of I that would directly impact on coverage and error of both algorithms, possibly reducing the actual performance differences. Moreover, tuning the number m of hash functions could outline a better trade-off between the accuracy of sketches and their size. Finally, we plan to extend our approach to different application domains.

7. REFERENCES

[1] R. Baumann, F. Legendre, and P. Sommer. Generic mobility simulation framework (gmsf). In *MobilityModels '08: Proceeding of the 1st ACM SIGMOBILE workshop on Mobility models*, pages 49–56, New York, NY, USA, 2008. ACM.

[2] L. Becchetti, P. Boldi, C. Castillo, and A. Gionis. Efficient semi-streaming algorithms for local triangle counting in massive graphs. In *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 16–24, New York, NY, USA, 2008. ACM.

[3] T. Bohman, C. Cooper, and A. M. Frieze. Min-wise independent linear permutations. *The Electronic Journal of Combinatorics,*, 7, 2000.

[4] A. Z. Broder. Identifying and filtering near-duplicate documents. In *Combinatorial Pattern Matching, 11th Annual Symposium, CPM 2000, Montreal, Canada, June 21-23, 2000, Proceedings*, volume 1848 of *Lecture Notes in Computer Science*, pages 1–10. Springer, 2000.

[5] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher. Min-wise independent permutations. In *ACM Symposium on the Theory of Computing*, New York, NY, USA, 1998.

[6] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig. Syntactic clustering of the web. In *Proc. of the World Wide Web Conference*, 1997.

[7] A. T. Campbell, S. B. Eisenman, N. D. Lane, E. Miluzzo, R. A. Peterson, H. Lu, X. Zheng, M. Musolesi, K. Fodor, and G.-S. Ahn. The rise of people-centric sensing. *IEEE Internet Computing*, 12:12–21, 2008.

[8] C. Cornelius, A. Kapadia, D. Kotz, D. Peebles, M. Shin, and N. Triandopoulos. Anonysense: privacy-aware people-centric sensing. In *MobiSys*, pages 211–224, 2008.

[9] European Commission Working Group Assessment of Exposure to Noise (WG-AEN). Good Practice Guide for Strategic Noise Mapping and the Production of Associated Data on Noise Exposure, January 2006.

[10] L. Filipponi, S. Santini, and A. Vitaletti. Data collection in wireless sensor networks for noise pollution monitoring. In *DCOSS '08: Proceedings of the 4th IEEE international conference on Distributed Computing in Sensor Systems*, pages 492–497, Berlin, Heidelberg, 2008. Springer-Verlag.

[11] E. Kanjo. Noisespy: A real-time mobile phone platform for urban noise monitoring and mapping. *Mobile Networks and Applications*.

[12] N. D. Lane, S. B. Eisenman, M. Musolesi, E. Miluzzo, and A. T. Campbell. Urban sensing systems: Opportunistic or participatory. In *In Proc. ACM 9th Workshop on Mobile Computing Systems and Applications (HOTMOBILE '08)*, 2008.

[13] N. Maisonneuve, M. Stevens, M. E. Niessen, P. Hanappe, and L. Steels. Citizen noise pollution monitoring. In *dg.o '09: Proceedings of the 10th Annual International Conference on Digital Government Research*, pages 96–103. Digital Government Society of North America, 2009.

[14] Metrosense - secure people-centric sensing at scale. http://metrosense.cs.dartmouth.edu/.

[15] N. Pham, R. Ganti, Y. Uddin, S. Nath, and T. Abdelzaher. Privacy-preserving reconstruction of multidimensional data maps in vehicular participatory sensing. In J. Silva, B. Krishnamachari, and F. Boavida, editors, *Wireless Sensor Networks*, volume 5970 of *Lecture Notes in Computer Science*, pages 114–130. Springer Berlin / Heidelberg, 2010.

[16] S. Zhong, L. (erran Li, Y. G. Liu, and Y. R. Yang. Privacy-preserving locationbased services for mobile users in wireless networks. Technical report, 2004.