A Framework of Energy Efficient Mobile Sensing for Automatic User State Recognition*

Yi Wang† wangyi@usc.edu
Jialiu Lin‡ jialiul@cs.cmu.edu
Murali Annavaram† annavara@usc.edu
Quinn A. Jacobson§ quinn.jacobson@nokia.com
Jason Hong‡ jasonh@cs.cmu.edu
Bhaskar Krishnamachari† bkrishna@usc.edu
Norman Sadeh‡ sadeh@cs.cmu.edu
†Ming Hsieh Department of Electrical Engineering, University of Southern California, Los Angeles, USA
‡School of Computer Science, Carnegie Mellon University, Pittsburgh, USA
§Nokia Research Center, Palo Alto, USA

ABSTRACT

Urban sensing, participatory sensing, and user activity recognition can provide rich contextual information for mobile applications such as social networking and location-based services. However, continuously capturing this contextual information on mobile devices consumes huge amount of energy. In this paper, we present a novel design framework for an Energy Efficient Mobile Sensing System (EEMSS). EEMSS uses hierarchical sensor management strategy to recognize user states as well as to detect state transitions. By powering only a minimum set of sensors and using appropriate sensor duty cycles EEMSS significantly improves device battery life. We present the design, implementation, and evaluation of EEMSS that automatically recognizes a set of users' daily activities in real time using sensors on an off-the-shelf high-end smart phone. Evaluation of EEMSS with 10 users over one week shows that our approach increases the device battery life by more than 75% while maintaining both high accuracy and low latency in identifying transitions between end-user activities.

Categories and Subject Descriptors

C.3.3 [**Special Purpose and Application Based Systems**]: Real-time and embedded systems

General Terms

Design, Experimentation, Measurement, Performance

*We'd like to acknowledge partial support for this work from Nokia Inc and National Science Foundation, numbered NSF CNS-0831545.

MobiSys'09, June 22–25, 2009, Kraków, Poland.

Keywords

Energy efficiency, Mobile sensing, EEMSS, Human state recognition

1. INTRODUCTION

As the number of transistors in unit area doubles every 18 months following Moore's law, mobile phones are packing more features to utilize the transistor budget. Increasing the feature set is mostly achieved by integrating complex sensing capabilities on mobile devices. Today's high-end mobile device features will become tomorrow's mid-range mobile device features. Current sensing capabilities on mobile phones include WiFi, Bluetooth, GPS, audio, video, light sensors, accelerometers and so on. As such the mobile phone is no longer only a communication device, but also a powerful environmental sensing unit that can monitor a user's ambient context, both unobtrusively and in real time.

On the mobile application development front, ambient sensing and context information [1] have become primary inputs for a new class of mobile cooperative services such as real time traffic monitoring [2], and social networking applications such as Facebook [3] and MySpace [4]. Due to the synergistic combination of technology push and demand pull, context aware applications are increasingly utilizing various data sensed by existing embedded sensors. By extracting more meaningful characteristics of users and surroundings in real time, applications can be more adaptive to the changing environment and user preferences. For instance, it would be much more convenient if our phones can automatically adjust the ring tone profile to appropriate volume and mode according to the surroundings and the events in which the users are participating. Thus we believe user's contextual information brings application personalization to new levels of sophistication. While user's context information can be represented in multiple ways, in this paper we focus on using user state as an important way to represent the context. User state may contain a combination of features such as motion, location and background condition that together describe user's current context.

A big hurdle for context detection, however, is the limited battery capacity of mobile devices. The embedded sensors in the mobile devices are major sources of power consumption.

For instance, a fully charged battery on Nokia N95 mobile phone can support telephone conversation for longer than ten hours, but our empirical results show that the battery would be completely drained within six hours if the GPS receiver is turned on, whether it can obtain GPS readings or not. Hence, excessive energy consumption may become a major obstacle to broader acceptance of context-aware mobile applications or services, no matter how useful the service may be. In mobile sensing applications, energy savings can be achieved by shutting down unnecessary sensors as well as carefully selecting *sensor duty cycles* (i.e., sensors will adopt periodic sensing and sleeping instead of being sampled continuously). In this paper, we define *sensor sampling duration* as the length of the time a sensor is turned ON for active data collection. We define *sensor sleeping duration* as the time a sensor stays idle. The sensing and sleeping durations, or sensor duty cycles, are generally referred to as *sensor parameters*.

To address the problem of energy efficiency in mobile sensing, we present the design, implementation, and evaluation of EEMSS, an energy efficient mobile sensing system that incorporates a hierarchical sensor management scheme for power management. EEMSS uses a combination of sensor readings to automatically recognize user state as described by three real-time conditions; namely motion (such as running and walking), location (such as staying at home or on a freeway) and background environment (such as loud or quiet). The core component of EEMSS is a sensor management scheme which defines user states and state transition rules by an XML styled state descriptor. This state descriptor is taken as an input and is used by our sensor assignment functional block to turn sensors on and off based on a user's current condition.

The benefits of our sensor management scheme are threefold. First, the state description mechanism proposed in this paper is a flexible way to add/update user states and their relationship to the sensors. For instance, to account for emerging application needs new states and sensors may be incrementally added to the state description. Second, to achieve energy efficiency, the sensor management scheme assigns the minimum set of sensors and heuristically determines sampling lengths and intervals for these set of sensors to detect user's state as well as transitions to new states. Lastly, our sensor management scheme can be easily extended as a middleware that manages sensor operations and provides contextual information to higher layer applications with multiple types of devices and sensors involved.

EEMSS is currently implemented and evaluated on Nokia N95 devices. In our EEMSS implementation, the state description subsystem currently defines the following states: "Walking", "Vehicle", "Resting", "Home_talking", "Home_ent ertaining", "Working", "Meeting", "Office_loud", "Place_quiet", "Place_speech" and "Place_loud". All these states are specified as a combination of built-in Nokia N95 sensor readings. The sensors used to recognize these states are accelerometer, WiFi detector, GPS, and microphone. EEMSS incorporates novel and efficient classification algorithms for real-time user motion and background sound recognition, which form the foundation of detecting user states. We have also conducted a field study with 10 users at two different university campuses to evaluate the performance of EEMSS. Our results show that EEMSS is able to detect states with 92.56% accuracy and improves the battery lifetime by over 75%, compared to existing results. Note that although in this paper we focus only on states that can be detected by integrated sensors on mobile devices, our sensor management scheme is general enough that one can apply our infrastructure to mobile sensing systems that involves more sensors and devices.

The remainder of this paper is organized as follows. In Section 2, we present relevant prior works and their relations to our study. In Section 3, we describe the sensor management scheme which is the core component of EEMSS. In Section 4, we introduce a case study of EEMSS on Nokia N95 devices and present the system architecture and implementation. In Section 5, we list the empirical results of different sensor power consumptions as one of the motivations of our system design and discuss the sensor duty cycling impact on system performance. In Section 6, we propose novel real-time activity and background sound classification mechanisms that result in good classification performance. The user study is presented in Section 7, where we evaluate our system in terms of state recognition accuracy, state transition discovery latency and device lifetime. Finally, we present the conclusion and our future work direction in Section 8.

2. RELATED WORK

There has been a fair amount of work investigating multi-sensor mobile applications and services in recent years. The concept of sensor fusion is well-known in pervasive computing. For example, Gellersen *et al.* [5] pointed out the idea that combining a diverse set of sensors that individually captures just a small aspect of an environment may result in a total picture that better characterizes a situation than location or vision based context.

Motion sensors have been widely used in monitoring and recognizing human activities to provide guidance to specific tasks [6, 7, 8]. For example, in car manufacturing, a context-aware wearable computing system designed by Stiefmeier *et al.* [6] could support a production or maintenance worker by recognizing the worker's actions and delivering just-in-time information about activities to be performed. A common low cost sensor used for detecting motion is the accelerometer. With accelerometer as the main sensing source, activity recognition is usually formulated as a classification problem where the training data is collected with experimenters wearing one or more accelerometer sensors in a certain period. Different kinds of classifiers can be trained and compared in terms of the accuracy of classification [9, 10, 11, 12]. For example, more than 20 human activities including walking, watching TV, running, stretching, etc. can be recognized with fairly high accuracy [12].

Most existing works to accurately detect user state require accelerometer sensor(s) to be installed on pre-identified position(s) near human body. Our aim is to avoid the use of obtrusive and cumbersome external sensors in detecting user state. As such, we remove the need to strap sensors to human body. EEMSS is able to accurately detect human states, such as walking, running and riding a vehicle by just placing the mobile phone anywhere on the user's body without any placement restrictions. In this context it is worth noting that Schmidt *et al.* [13] first proposed incorporating low level sensors to mobile PDAs/phones to demonstrate situational awareness. Several works have been conducted thereafter by using the commodity cell phones as sensing.

computing or application platforms [14, 15, 16, 17, 18, 19]. For example, "CenceMe" [16] enables members of social networks to share their sensing presence with their "buddies" in a secure manner. The system uses the integrated as well as external sensors to capture the users' status in terms of activity, disposition, habits and surroundings. A CenceMe prototype has been made available on Facebook, and the implementation and evaluation of the CenceMe application has also been discussed [17]. Similarly, "Sensay" [15] is a context-aware mobile phone and uses data from a number of sources to dynamically change cell phone ring tone, alert type, as well as determine users' "un-interruptible" states. "Sensay" requires input from an external sensor box which is mounted on the user's hip area and the system design does not have energy efficiency concern. Moreover, the decision module of "Sensay" is implemented on a computer instead of mobile device. In comparison, our approach in EEMSS design uses the off-the-shelf mobile device and manage sensors in a way such that sensing is conducted in an energy efficient manner.

Researchers from different fields have studied and used a large number of sensors including GPS, Bluetooth, WiFi detector, blood oxygen saturation sensor, accelerometer, electrocardiograph sensor, temperature sensor, light sensor, microphone, camera, etc. in projects such as urban/paticipatory sensing [14, 20, 21], activity recognition[22, 23, 24], and health monitoring [25, 26, 27]. For example, Whitesell *et al.* [21] have designed and implemented a system that analyzes images from air sensors captured from mobile phones and indoor air pollution information has been extracted by comparing the data to a calibrated chart. Targeting obesity problem in health monitoring domain, Annavaram *et al.* [24] showed that by using data from multiple sensors and applying multi-modal signal processing, seemingly similar states such as sitting and lying down can be accurately discriminated, while using only a single accelerometer sensor these states can not be easily detected. Wu *et al.* [27] have designed "SmartCane" system which provides remote monitoring, local processing, and real-time feedback to elder patients in order to assist proper usage of canes to reduce injury and death risks. While these works only focused on how to more accurately detect human context using one or more sensors, in this paper we emphasize both energy efficiency and state detection accuracy. In fact, in [17], the authors were well aware of the battery life constraint of mobile devices and different duty cycling mechanisms have been considered and tested for different physical sensors. However the lack of intelligent sensor management method still withholds the device lifetime by a significant amount.

The problem of energy management on mobile devices has been well-explored in the literature such as [28, 29, 30, 31, 32]. For example, Viredaz *et al.* [28] surveyed many fundamental but effective methods for saving energy on handheld devices in terms of improving the design and cooperation of system hardware, software as well as multiple sensing sources. Event driven power-saving method is investigated by Shih *et. al.* to reduce system energy consumptions [31]. In their work, the authors focused on reducing the *idle power*, the power a device consumes in a "standby" mode, such that a device turns off the wireless network adaptor to avoid energy waste while not actively used. The device will be powered on only when there is an incoming or outgoing call or when the user needs to use the PDA for other purposes. To further explore the concept of event-driven energy management, a hierarchical power management method was used in [32]. In their demo system "Turdecken", a mote is used to wake up the PDA, which in turn wakes up the computer by sending a request message. Since the power required by the mote is enough for holding the whole system standby, the power consumption can be saved during system idle time.

In our system design, we build on many of these past ideas and integrate them in the context of effective power management for sensors on mobile devices. In order to achieve human state recognition in an energy efficient manner, we have proposed a hierarchical approach for managing sensors, and do so in such a way that still maintains accuracy in sensing the user's state. Specifically, power hungry sensors are only activated whenever triggered by power efficient ones. By only duty cycling the minimum set of sensors to detect state transition and activating more expensive ones on demand to recognize new state, the device energy consumption can be significantly reduced. A similar idea was explored by the "SeeMon" system [33], which achieves energy efficiency by only performing context recognition when changes occur during the context monitoring. However, "SeeMon" focuses on managing different sensing sources and identifying condition changes rather than conducting people-centric user state recognition.

3. SENSOR MANAGEMENT METHODOLOGY

In this section we will describe our design methodology for EEMSS framework. The core component of EEMSS is a *sensor management scheme* which uniquely describes the features of each user state by a particular sensing criteria and state transition will only take place once the criteria is satisfied. An example would be that "meeting in office" requires the sensors to detect both the existence of speech and the fact that the user is currently located in office area. EEMSS also associates the set of sensors that are needed to detect state transitions from any given state. For example, if the user is "sitting still" and in order to detect "movement" mode accelerometer must be sampled periodically.

3.1 State and Sensor Relationship

Sensor assignment is achieved by specifying an XML-format state descriptor as system input that contains all the states to be automatically classified as well as sensor management rules for each state. The system will parse the XML file as input and automatically generate a sensor management module that serves as the core component of EEMSS and controls sensors based on real-time system feedback. In essence, the state descriptor consists of a set of state names, sensors to be monitored, and conditions for state transitions. It is important to note the system designer must be well familiar with the operation of each sensor and how a user state can be detected by a set of sensors. State description must therefore be done with care so as to not include all the available sensors to detect each state since such a gross simplification in state description will essentially nullify any energy savings potential of EEMSS.

Figure 1 illustrates the general format of a state descriptor and the corresponding state transition process. It can be seen that a user state is defined between the "<State>"

```
<StateDescriptor>
  <State>
    <StateName> state1 </StateName>
    <Sensor>
      <SensorName> sensor1 </SensorName>
      <Case>
        <Condition> sensor reading 1</Condition>
        <NextState> State2 </NextState>
      </Case>
      <Case>
       .
       .
      </Case>
    </Sensor>
    <Sensor>
     .
     .
    </Sensor>
  </State>
  <State>
   <StateName> state2 </StateName>
    <Sensor>
      <SensorName> sensor2 </SensorName>
      <Case>
        <Condition> sensor reading 2</Condition>
        <Sensor>
          <SensorName> sensor3 </SensorName>
          <Case>
            <Condition> sensor reading 3</Condition>
            <NextState> State1 </NextState>
          </Case>
          <Case>
            <Condition> sensor reading 4</Condition>
            <NextState> State3 </NextState>
          </Case>
        </Sensor>
      </Case>
      <Case>
       .
       .
      </Case>
    </Sensor>
    <Sensor>
     .
     .
    </Sensor>
  </State>
 .
 .
</StateDescriptor>
```

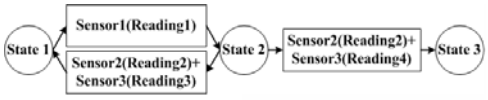

Figure 1: The format of XML based state descriptor and its implication of state transition.

and "</State>" tags. For each state, the sensor(s) to be monitored are specified by "<Sensor>" tags. The hierarchical sensor management is achieved by assigning new sensors based on previous sensor readings in order to detect state transition. If the state transition criteria has been satisfied, the user will be considered as entering a new state (denoted by "<NextState>" in the descriptor) and the sensor management algorithm will restart from the new state. For example, based on the sample description in Figure 1, if the user is at "State2" and "Sensor2" returns "Sensor reading 2" which is not yet sufficient for detecting state transition, "Sensor3" will be turned on immediately to further detect the user's status in order to identify state transition.

There are three major advantages of using XML as the format of state descriptor. First, XML is a natural language to represent states in a hierarchical fashion. Second, new state descriptors can be added and existing states can be modified with relative ease even by someone with limited programming experience. Finally, XML files are easily parsed by modern programming languages such as Java and Python thereby making the process portable and easy to implement.

3.2 Setting Sensor Duty Cycles

Recall that in the first phase of state description the system designer will specify the list of states and the sensors that are required to detect that state and all the possible state transitions. In the second phase, the system designer must carefully set the sampling period and duty cycles to balance the state detection accuracy with energy efficiency. In our current implementation these values are set manually based on experimentation. In this phase of system configuration we also design and test classification algorithms that recognize user status based on different sensor readings. These classification algorithms are pre-trained based on extensive experiments conducted by researchers. We will present the specific sensor parameters used in EEMSS in Section 5 and the classification algorithms in Section 6.

3.3 Generalization of the Framework

We would like to emphasize that the system parameters need only to be set once after the training phase and can be used repeatedly during the operation of the sensing system. However, we do recognize that the process of manually setting sensor duty cycles for all sensors and states may be cumbersome even if it is rare. We believe there are ways to semi-automate sensor assignment mechanism. In order to provide an automated sensor assignment mechanism rather than manually specifying sensor parameters, a sensor information database could be built *a priori* on each mobile device that stores the sensor power consumption statistics and also how the data provided by one sensor can be approximated with the data from a different sensor. For instance, position data from GPS can be approximated using cell tower triangulations. We envision that in future the sensor management effort will be pushed from the developer-end to the device-end where the sensor information database serves as a stand-alone sensor management knowledge center. In this scenario the sensor management scheme as well as the sensor sampling parameters could be generated or computed based on knowledge database with limited human input.

As noted earlier our XML based state description mechanism is highly scalable as new states can be added or updated easily. With each new state addition in our current implementation we need to define a classification algorithm that recognizes the new state. Once the classification algorithm is defined we can generate the sensor parameters after a brief training period.

Various sensors makes the user's contextual information available in multiple dimensions, from which a rich set of user states can be inferred. However, in most cases different users or higher layer applications may only be interested in identifying a small subset of states and exploit the state information for application customization. For example, a ring tone adjustment application, which can automatically adjust the cell phone alarm type, may only need to know the property of background sound in order to infer the current situation. A medical application may require the system to monitor one's surrounding temperature, oxygen level and the user's motion such as running and walking to give advise to patient or doctors. In a personal safety application, one factor that one may care is whether the user is riding a vehicle or walking alone such that the mobile client is able to send warning messages to the user when he or she is detected walking in an unsafe area at late night. These are all examples of mobile sensing systems with particular needs, by which our framework design can be potentially adopted.

4. EEMSS IMPLEMENTATION – A CASE STUDY

4.1 Description

In this section we will describe a practical implementation of a state detection system using EEMSS framework. For this case study we focus on using only built-in sensors on Nokia N95 device to detect states. N95 has several built-in sensors, including GPS, WiFi detector, accelerometer, and embedded microphone. The goal of the case study is to conduct a prototype implementation using EEMSS framework and to quantify the performance in terms of state recognition accuracy, detection latency, as well as energy efficiency. As such we select a set of states that describe the user's daily activities and have defined the state and sensor relationships in XML using the format introduced in Section 3. Table 1 illustrates the set of user states to be recognized by EEMSS and three characteristic features that define each of these states. The three features are the location, motion and background sound information. The list of sensors necessary to detect these three features are also shown in Table 1. We selected a sample set of user states that can all be detected solely using the in-built sensors on N95 in this case study.

For each user state, our EEMSS implementation monitors the characteristic features defining that state by reading a corresponding sensor value. For instance, various background sounds can be detected and discriminated by sampling the microphone sensor. In addition to monitoring the current state, EEMSS also monitors a set of sensors that define a state transition. Recall that state description using hierarchical sensor management not only defines the set of sensors to be sampled, but also specifies possible state transitions and the sensor readings that trigger the transition. If a state transition happens, a new set of sensors will be turned on to recognize one's new activity. Here we select one of the user states (Walking) and illustrate how the state transition is detected when the user is walking outdoor. Figure 2 shows the hierarchical decision rules. It can be seen that the only sensor that is being periodically sampled is GPS when the user is walking, which returns both the Geocoordinates and the user's speed information that can be used to infer user's mode of travel. If a significant amount of increase is found on both user speed and recent distance of travel, a state transition will happen and the user will be considered riding a vehicle. Once GPS times out due to lost of satellite signal or because the user has stopped moving for a certain amount of time, a WiFi scan will be performed to identify the current place by checking the surrounding wireless access points. Note that the wireless access point sets for one's frequently visited places such as home, cafeteria, office, gym, etc. can be pre-stored on the device. Finally, the background sound can be further sensed based on the audio signal processing. We will quantify the accuracy and device energy efficiency in Section 7.

It is important to note that the Nokia N95 device contains more sensors such as Bluetooth, light sensor, and camera. However, we chose not to use these sensors in current EEMSS case study implementation due to either low technology penetration rate or sensitivity to the phone's physical placement. For instance, experiments have been conducted where a mobile device will probe and count the neighboring Bluetooth devices, and the results show that the number of such devices discovered is very low (usually less than 5),

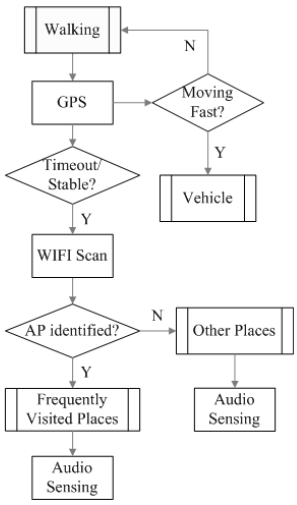

Figure 2: The sequential sensor management rules used to detect state transitions when the user is walking outdoors.

even though a big crowd of people is nearby. Light sensor is also not used in our study because the result of light sensing depends highly on whether the sensor can clearly see the ambient light or its view is obstructed due to phone placement in a pocket or handbag. Therefore it could potentially provide high percentage of false results. Moreover, since we focus on an automated real-time state recognition system design, the camera is also not considered as part of our study since N95 camera shutter requires manual intervention to turn on and off the camera. Even though these sensors have not been used in our case study, they still remain as important sensing sources for our future study.

4.2 Architecture and Implementation

The main components of EEMSS, including sensor management and activity classification, have been implemented on J2ME on Nokia N95 devices. The popularity of Java programming and the wide support of J2ME by most of the programmable smart phone devices ensure that our system design achieves both portability and scalability. However, the current version of J2ME does not provide APIs that allow direct access to some of the sensors such as WiFi and accelerometer. To overcome this, we created a Python program to gather and then share this sensor data over a local socket connection.

The system can be viewed as a layered architecture that consists of a sensor management module, a classification module, and a sensor control interface which is responsible of turning sensors on and off, and obtaining sensed data. We also implemented other components to facilitate debugging and evaluation, including real-time user state updates, logging, and user interfaces. Figure 3 illustrates the design of the system architecture and the interactions among the components.

As mentioned in the previous subsection, the sensor management module is the major control unit of the system. It first parses a state description file that describes the sensor management scheme, and then controls the sensors based on the sensing criteria of each user state and state transition conditions by specifying the minimum set of sensors to

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.