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.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
MobiSys'09, June 22–25, 2009, Kraków, Poland.
Copyright 2009 ACM 978-1-60558-566-6/09/06 ...$5.00.

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>
```

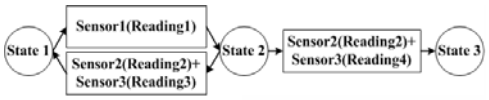

Figure 1: The format of XML based state descriptor and its implication of state transition.

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),

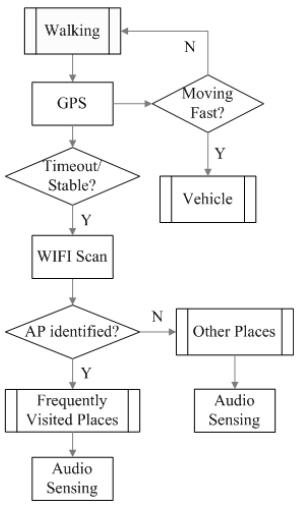

Figure 2: The sequential sensor management rules used to detect state transitions when the user is walking outdoors.

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

State Name	State Features			Sensors Monitored
	Location	Motion	Background Sound	
Working	Office	Still	Quiet	Accelerometer, Microphone
Meeting	Office	Still	Speech	Accelerometer, Microphone
Office_loud	Office	Still	Loud	Accelerometer, Microphone
Resting	Home	Still	Quiet	Accelerometer, Microphone
Home_talking	Home	Still	Speech	Accelerometer, Microphone
Home_entertaining	Home	Still	Loud	Accelerometer, Microphone
Place_quiet	Some Place	Still	Quiet	Accelerometer, Microphone
Place_speech	Some Place	Still	Speech	Accelerometer, Microphone
Place_loud	Some Place	Still	Loud	Accelerometer, Microphone
Walking	Keep on changing	Moving Slowly	N/A	GPS
Vehicle	Keep on changing	Moving Fast	N/A	GPS

Table 1: The states and their features captured by our system (EEMSS).

be monitored under different scenarios (states). The sensor management module configures the sensors in real-time according to the intermediate classification result acquired from the classification module and informs the sensor control interface what sensors to be turned on and off in the following step.

In our case study, the classification module is the consumer of the sensor raw data. The classification module first processes the raw sensing data into desired format. For example, the magnitude of 3-axis accelerometer sensing data is computed, and FFT is performed on sound clips to conduct frequency domain signal analysis. The classification module returns user activity and position feature such as "moving fast", "walking", "home wireless access point detected" and "loud environment" by running classification algorithms on processed sensing data. The resulting user activity and position information are both considered as intermediate state which will be forwarded to the sensor management module. The sensor management module then determines whether the sensing results satisfy the sensing criteria and decides sensor assignments according to the sensor management algorithm.

The sensor interface contains APIs that provide direct access to the sensors. Through these APIs, application can obtain the sensor readings and instruct sensors to switch on/off for a given duty cycle, as well as change the sample rate. As mentioned previously, due to J2ME limitations, GPS and embedded microphone are operated through J2ME APIs while accelerometer and WiFi detector are operated through Python APIs.

5. ENERGY CONSUMPTION MEASUREMENT AND SENSOR DUTY CYCLES

In this section, we present our methodology to determine the energy consumption of sensors used in the current EEMSS case study, in order to understand how to best coordinate them in an effective way. We conducted a series of power consumption measurements on different built-in sensors that are used in this case study, including GPS, WiFi detector, microphone and accelerometer. We discuss the implementation of duty cycling mechanisms on the sensors and the corresponding energy cost for each sensor sampling.

The sensors on a mobile phone can be categorized into two classes. The first class includes the accelerometer and microphone. These sensors once turned on operate continuously and require an explicit signal to be turned off. More-

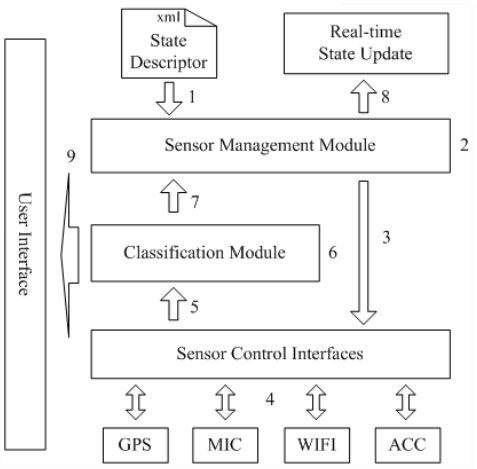

Figure 3: System architecture of EEMSS implementation on Nokia N95. (1)System reads in the XML state descriptor which contains the sensor management scheme. (2)Management module determines the sensors to be monitored based on current user state which is specified by the sensor management scheme. (3)Management module instructs the sensor control interface to turn on/off sensors. (4) Sensor control interface operates individual sensors. (5) Sensor interface reports readings to classification module. (6)Classification module determines the user state. (7)Classification module forwards the intermediate classification result to management module. (8) The user's state is updated and recorded in real-time. (9) The relevant information is also displayed on the smart phone screen.

Sensor	Power(W)	Current(A)
First Class		
Accelerometer	0.0599	0.0150
Microphone	0.2786	0.0707
Second Class		
GPS	0.3308	0.0862
WiFi Scan	1.4260	0.3497
Bluetooth Scan	0.1954	0.0486

Table 2: Power and current consumption summary for different sensors on Nokia N95.

Sensor	Duty Cycles	Computation Time/Sample	Energy(J)/Sample
Accelerometer	6s sensing + 10s sleeping	< 0.1s	0.359
Microphone	4s sensing + 180s sleeping	Quiet: < 0.5s. Loud/Speech: ~ 10s.	1.114
GPS	Queries every 20s, timeout in 5 minutes	< 0.1s	6.616
WiFi scan	Event triggered (< 2s to finish)	< 0.1s	2.852

Table 3: Sensor duty cycles, device computation time and sensor energy cost per sample.

over, both the accelerometer and the microphone need to be activated for a minimum period of time to obtain meaningful sensing data. For instance, collecting an instant audio sample does not provide any meaningful data to represent the background sound type. The second class of sensors includes GPS, WiFi detector, and Bluetooth scanner. These sensors when turned on gather instantaneous samples, and are automatically turned off when the sampling interval is over.

For both classes, the energy cost of sensing depends not only on the instant power drain, but also on the operating duration of the sensors. For example, due to API and hardware limitations, the GPS on Nokia N95, even when using assisted-GPS functionality, requires at least 10 seconds to successfully synchronize with satellites and will remain active for about 30 seconds after a location query. As such, the overall energy consumption even to collect a single GPS sample is quite significant. A WiFi scan takes less than 2 seconds to finish, and a Bluetooth scan takes around 10 seconds to complete, with the duration increasing linearly with the number of Bluetooth devices in the neighborhood.

5.1 Power Consumption Measurement

We first measure sensor power consumptions through Nokia Energy Profiler [34], a stand-alone application that allows developers to test and monitor application energy usage in real time. Measurement results are summarized in Table 2. From these results, it can be seen that power consumed by different sensors vary greatly. Among these sensors, accelerometer consumes the least amount of power compared to other sensors, and fortunately accelerometer is also capable of detecting any body movements with a high precision. Hence accelerometer could be first indicator of state transition with high probability if it involves user body motion. In such a case, accelerometer could be sampled periodically as triggers to invoke other sensors if necessary. On the other hand, due to the large power drain and long initialization time, GPS is used only when it is necessary to measure the speed of user's movement so as to discriminate between modes of travel such as riding in vehicle versus walking.

5.2 Sensor Duty Cycles, Computation Time and Energy Cost

EEMSS achieves its energy efficiency goals using a two pronged approach. First, the state descriptors guarantee that only a minimum set of sensors are monitored in any given state. Second, energy consumption is reduced by carefully assigning duty cycle to each sensor. Note that duty cycling a sensor is going to tradeoff reduced energy consumption with potentially reduced accuracy/speed of state detection. In our current implementation we manually set these duty cycles by running extensive trials in our training phase.

Table 3 summarizes the duty cycles for each of the four sensors implemented in EEMSS. It can be seen that accelerometer and microphone sensing both perform duty cycling where the sensor will be turned on and off repeatedly based on the parameters shown in Table 3. Note that even though the energy cost can be saved by reducing sensing intervals, if the sampling period is too short the sensor readings will not be sufficient to represent the real condition. On the other hand, while a longer sensing period could increase the robustness of state recognition, it would also consume more energy. The same tradeoff applies for sleep interval as well. A longer sleep interval may reduce energy consumption, but the detection latency will be increased. There are two reasons for assigning longer duty cycles to the microphone versus the accelerometer, as indicated by the parameters in Table 3. First, the accelerometer draws significantly less power, and hence it can be sampled more frequently with small impact on battery lifetime. Second, the accelerometer captures user motion change, which tolerates less detection delay compared to identifying background sound type.

GPS is queried periodically when the user is moving outdoors, to provide location and speed information. We allow 5 minutes time out interval for GPS, a relatively long duration for the GPS to lock satellite signal. We found in our experiments that under some circumstances (e.g.: when the user is walking between two tall buildings or taking a bus), the N95 GPS may be either temporarily unavailable or needs a much longer time than usual to acquire the signal. Therefore, a longer timeout duration is required for the GPS to successfully get readings. WiFi scanning is event-based rather than being performed periodically. In EEMSS, a WiFi scan is performed under two scenarios: (1) when the user is detected as moving, a WiFi scan is conducted to check if the user has left his or her recent range, and (2) when the user has arrived at a new place, we compare the nearby wireless access points set with known ones in order to identify the user's current location.

Even though the duty cycle parameters have been refined through extensive empirical tests, the sensing parameters finally adopted by EEMSS (as shown in Table 3) may not achieve the optimal tradeoff between energy consumption and state detection accuracy. In our current implementation, the parameters are manually tuned and each sensor follows a fixed sampling rate when activated. No optimization or dynamic adjustment has been implemented. In future we plan to construct models that capture the tradeoff between energy and state detection accuracy, and find automatic ways to set the sensing parameters to achieve better tradeoff. It is also likely that the sensing parameters may be need be readjusted dynamically based on the real time results.

The computation time (including the time for data processing and classification) and sensor energy consumed per sample based on sensor duty cycle parameters, are summarized in Table 3. It can be seen that except for loud au-

dio signal processing and classification, which takes approximately 10 seconds to complete (mainly consumed at the FFT stage), all other computations finish almost instantaneously, which enables our system to conduct real-time state recognition. The energy consumption results not only prove the fact that shutting down unnecessary sensing is important, but also provide us useful insights on designing optimal duty cycles in future work.

6. SENSOR INFERENCE AND CLASSIFICATION

In this section, we discuss the sensing capabilities and potential human activities that could be inferred from sensors used in our case study. We also discuss our proposed classification algorithms to detect user states of interest.

6.1 GPS Sensing and Mode of Travel Classification

In our case study the primary purpose of using GPS is to detect user's mode of travel. Besides providing real-time location tracking, GPS can also provide user's velocity at a given instance. By combining the instantaneously velocity information and the recent distance of travel measured by comparing current position with previous ones it is possible to robustly distinguishing one's basic mode of travel such as walking or riding a vehicle. For example, if the velocity is greater than 10 mph we consider that the user is using an automotive transport. The classifier is trained by several location tracking records of user and certain threshold values are identified and implemented into the classification algorithm.

GPS can also be used to identify when a user has entered a building or other indoor environment since a location request timeout will occur since the satellite signals are are likely to be blocked in the indoor environment. It is worth mentioning that from the system implementation point of view, obtaining instant speed as well as the location request timeout functionality are both supported by J2ME API.

6.2 WiFi Scanning and Usage

The MAC address of visible wireless access points around the user can be inferred by performing a WiFi scan. Since MAC address of each wireless access point is unique, it is possible to tag a particular location by the set of access points in that location. Therefore the mobile device is able to automatically identify its current location by simply detecting nearby wireless access points. For example, it is easy to tell that the user is at home if the WiFi scan result matches his or her home access point set that is premorized by the device. In our current EEMSS implementation, the wireless access points feature of the user's home and office (if applicable) will be pre-recorded for recognition purpose. While in our current implementation we pre-record the set of access points for each of user's well defined locations, such as home and office, there are several alternative implementations such as SkyHook [35] that provide the location information by a database table lookup of a set of access points.

WiFi scan can also be used to monitor a user's movement range since a wireless access point normally covers an area of radius 20-30m. Hence if the user moves out of his/her recent range a WiFi scan will detect that the current set of WiFi access points are replaced by a new one. In our system implementation, if the user is detected moving continuously by accelerometer, a WiFi scan will be performed to check whether the user has left his or her recent location and if so, GPS will be turned on immediately to start sampling location information and classify the user's mode of travel.

Mode	STDV Range of Magnitude
Still	0 - 1.0334
Walk	9.2616 - 21.3776
Run	35.3768 - 52.3076
Vehicle	4.0204 - 12.6627

Table 4: Standard deviation range of accelerometer magnitude readings for different user activities

	Still	Vehicle	Walking	Running
Still	99.44%	0.56%	0	0
Vehicle	8.81%	73.86%	16.29%	1.04%
Walking	1.18%	10.62%	88.20%	0
Running	0	0	0	100%

Table 5: Classification results based on standard deviation of accelerometer magnitude values. The first column represents the ground truth while the first row represents the classification results based on accelerometer readings.

6.3 Real-time Motion Classification Based on Accelerometer Sensing

Activity classification based on accelerometer readings has been widely studied using various machine learning tools [9, 10, 11, 12]. However, in most of the previous works one or more accelerometer sensors have to be attached to specific body joints such as knees and elbows. Several data features are then extracted from readings of multiple accelerometers in order to design sophisticated classifiers to recognize user activities. Most of these classification algorithms are both data and compute intensive and hence are unsuitable for real-time classification using current mobile phone computing capabilities.

In our system design, mobile phone is the only source of accelerometer readings. We only make the assumption that the mobile phone is carried by the user at all times without any placement restrictions. Hence, it becomes extremely difficult to perform motion classification using accelerometers alone as is done in previous study [12]. We use only the standard deviation of accelerometer magnitude as one of the defining features independent of phone placement in order to conduct real-time motion classification.

We have collected accelerometer data in 53 different experiments distributed in two weeks in order to train the classifier. The lengths of experiment vary from several minutes to hours. Within each empirical interval, the person tags the ground truth of his/her activity information for analysis and comparison purposes. The standard deviation for different activities within each empirical interval is computed off-line. Table 4 shows the range of standard deviation distribution based on different data sets collected.

It can be found out that there exist certain standard deviation threshold values that could well separate stable, walk-

ing, running, and vehicle mode with high accuracy. In order to verify this observation, we have implemented a real-time motion classification algorithm on N95 mobile phone that compares the standard deviation of accelerometer magnitude values with the thresholds in order to distinguish the user's motion. The device is carried by the user without explicit requirement of where the phone should be placed. 26 experiments have been conducted each containing a combination of different user motions. The standard deviation of accelerometer magnitude is computed every 6 seconds, and right after which the user's motion is being classified. Table 5 shows the classification results in percentage of recognition accuracy. It can be seen that the algorithm works very well for extreme conditions such as stable and running. Furthermore, even though the classifier tends to be confused with walking and vehicle mode due to feature overlap, the accuracy is still well maintained above 70%.

In our case study of EEMSS, since we do not explicitly require the system to identify states such as "Running" and that GPS is already sufficient to distinguish the mode of travel states including "Walking" and "Vehicle" as described in Section 6.1, the accelerometer is simply used to trigger other sensors such as WiFi detector whenever user motion is detected. The accelerometer will only be turned on as classification tool of user motion when the GPS becomes unavailable. However, note that the framework design of EEMSS is general enough that allows one to specify new states such as "Running" in the XML state descriptor as well as the corresponding sensor management rule (e.g.: accelerometer classification is required). The state descriptor will be parsed and understood by the system which in turn makes sensor control decisions accordingly.

6.4 Real-time Background Sound Recognition

This subsection describes the algorithm used for background sound classification. These algorithms were coded in Java and run on N95 to classify sound clips recorded using N95. The device records a real time audio clip using microphone sensor and the recorded sound clip will go through two classification steps (Figure 4). First, by measuring the energy level of the audio signal, the mobile client is able to identify if the environment is silent or loud. Note that the energy E of a time domain signal $x(n)$ is defined by $E = \sum_n |x(n)^2|$. Next, if the environment is considered loud, both time and frequency domains of the audio signal are further examined in order to recognize the existence of speech. Specifically, speech signals usually have higher silence ratio (SR) [36] (SR is the ratio between the amount of silent time and the total amount of the audio data) and significant amount of low frequency components. If speech is not detected, the background environment will simply be considered as "loud" or "noisy", and no further classification algorithm will be conducted to distinguish music, noise and other types of sound due to their vast variety of the signal features compared to speech.

SR is computed by picking a suitable threshold and then measuring the total amount of time domain signal whose amplitude is below the threshold value. The Fast Fourier Transform has been implemented such that the mobile device is also able to conduct frequency domain analysis to the sound signal in real time. Figure 5 shows the frequency domain features of four types of audio clips, including a male's speech, a female's speech, a noise clip and a music clip. It

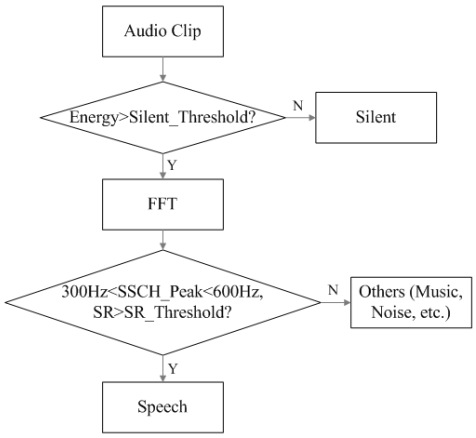

Figure 4: Decision tree based background sound classification algorithm.

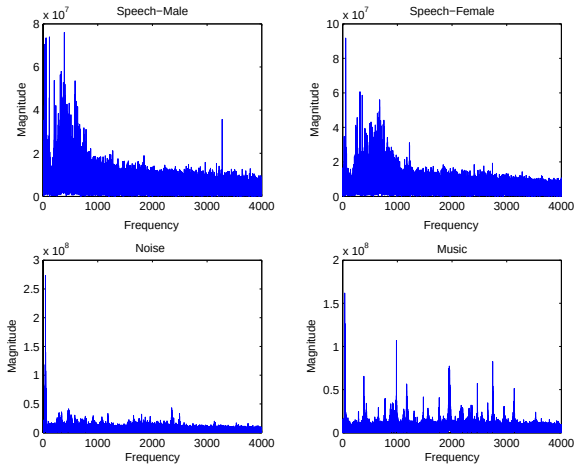

Figure 5: Comparison of frequency domain features of different audio signals.

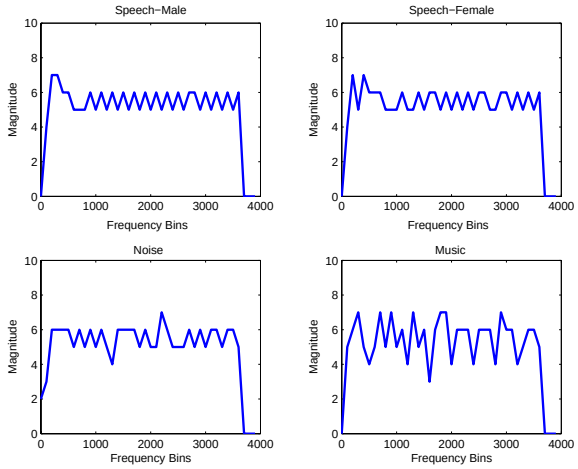

Figure 6: Frequency histogram plots after applying SSCH to sound clips described in Figure 5.

	Speech	Music	Noise
$SR_{thres} = 0.6$	95.31%	18.00%	26.07%
$SR_{thres} = 0.7$	91.14%	16.00%	18.17%
$SR_{thres} = 0.8$	70.91%	10.00%	11.66%
$SR_{thres} = 0.9$	32.64%	8.00%	8.59%

Table 6: Percentage of sound clips classified as "speech" for different SR_{thres} values.

can be seen clearly that as compared to others, speech signals have significantly more weight on low frequency spectrum from 300Hz to 600Hz. In order to accomplish speech detection in real time, we have implemented the SSCH (Subband Spectral Centroid Histogram) algorithm [37] on mobile devices. Specifically, SSCH passes the power spectrum of the recorded sound clip to a set of highly overlapping bandpass filters and then computes the spectral centroid [1] on each subband and finally constructs a histogram of the subband spectral centroid values. The peak of SSCH is then compared with speech peak frequency thresholds (300Hz - 600Hz) for speech detection purpose. Figure 6 illustrates the outputs of applying SSCH algorithm to the sound clips shown in Figure 5. It can be found out clearly that the histogram peaks closely follow the frequency peaks in the original power spectrum.

The classification algorithm is trained and examined on the same data set including 1085 speech clips, 86 music clips and 336 noise clips, all with 4 seconds length which are recorded by Nokia N95 devices. We investigate the effect of different SR thresholds (denoted by SR_{thres}) on classification accuracy. The results of speech detection percentage are shown in Table 6. It can be seen that as SR threshold increases, the number of false positive results are reduced with sacrifice of speech detection accuracy. We choose $SR_{thres} = 0.7$ throughout our study which provides more than 90% of detection accuracy and less than 20% false positive results. The above classification results show that a 4-second sample of audio clip is long enough for the classifier to identify the background sound type. It is also important to note that the complexity of the SSCH algorithm is $O(N^2)$ and as the filter overlaps are small, the running time is empirically observed to be close to linear. Empirical results show that on average the overall processing time of a 4 seconds sound clip is lower than 10 seconds on N95 devices. In future as compute capabilities of mobile phones increase we expect the latency of such complex audio processing will be reduced significantly.

7. PERFORMANCE EVALUATION

7.1 Method

In this section, we present an evaluation of EEMSS, assessing its effectiveness in terms of state recognition accuracy, state transition detection latency, as well as energy efficiency.

We have conducted a user trial in November 2008 at University of Southern California and Carnegie Mellon University. The main purpose of the user trial was to test the performance of EEMSS system in a free living setting. We recruited 10 users from both universities including undergraduate and graduate students, faculties and their family. The recruitment drive was conducted through online mailing lists and flyers. Each participant was provided with a Nokia N95 device with the EEMSS framework pre-installed. Basic operating instructions were provided to the users at the start of the experimental trials. Each user carried the device with EEMSS running for no less than two days. Each participants was requested to fully charge the mobile battery before starting the EEMSS application. EEMSS will continue to run in the background till the battery is completely drained. Participants then fully charge the phone once again before starting the EEMSS application. The cycle of fully charging and discharging continued till the expiration of user's participation time in the experiments.

EEMSS automatically records the predicted user state using the three discriminating features: motion, location and background sound. For each state transition EEMSS recorded the new user state and the time stamp of when the user entered that state. The predicted user state data is stored locally on the mobile phone. In addition to carrying the mobile phone each user was also given a diary in order to manually record ground truth for evaluation purpose. The diary was made of a standardized booklet containing a table with fine-grained time line entries. Each entry of the booklet contains three simple questions. In particular, we asked participants to record their motion (e.g.: walking, in vehicle, etc), location, and background sound condition (e.g.: quiet, loud, speech, etc). We then compared diary entries with the EEMSS recognition results. There are two reasons mobile devices were not used to collect ground truth. First, in order to guarantee instantaneous state transition detection sensors need to be monitored continuously which leads to a significant reduction on device lifetime. Second, the device doesn't necessarily provide 100% state recognition accuracy due to classification algorithm constraints. Hence, we decided to use the simplest way where users wrote down their activities in the given booklet. At the end of the EEMSS evaluation period, we collected more than 260 running hours of data with more than 300 user state transitions detected.

7.2 Results

7.2.1 EEMSS Capabilities

EEMSS is able to characterize a user's state by time, location and activities. Besides providing real-time user state update, EEMSS keeps tracking the user's location by recording the Geo-coordinates of the user when he or she is moving outdoor (recall that GPS is turned on continuously and the location information is retrieved every 20 seconds in this scenario). Figure 7 and 8 visualize the data collected by EEMSS on 2-D maps. They show the daily traces captured by EEMSS of two different participants from CMU and USC on two campus maps respectively. Within the time frame of these two traces, the EEMSS application keeps running and the phone has not been recharged. Both users have reported that they took buses to school, and the dashed curves which indicate "Vehicle" mode are found to match the bus routes perfectly. The solid curves indicating "Walking" state match the traces that the user is walking between home and bus station, and within university campus.

[1]The spectral centroid C of a signal is defined as the weighted average of the frequency components with magnitudes as the weights: $C = \frac{\sum_f f \cdot X(f)}{\sum_f X(f)}$

Figure 7: Recognized daily activities of a sample CMU user. The figure shows the time, user location as well as background sound condition detected by EEMSS.

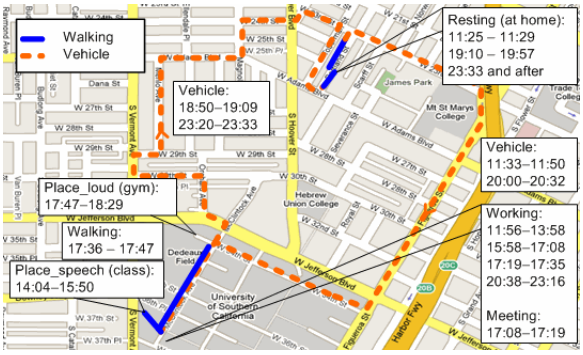

Figure 8: Recognized daily activities of a sample USC user. The figure shows the time, user location as well as background sound condition detected by EEMSS.

Besides monitoring location change and mode of travel of the user in real-time, EEMSS also automatically detects the surrounding condition when the user is identified to be still at some places in order to infer the user's activities. In Figure 7 and 8, by probing background sound, the surrounding conditions of the user can be classified as quiet, loud and containing speech. Consequently, the system is able to infer that the user is working, meeting, resting, etc. by combining the detected background condition with location information obtained by WiFi scan. Hence, we conclude that the user state information recognized by EEMSS accurately matched the ground truth as recorded by the users.

7.2.2 *State Recognition Accuracy*

We first present the state recognition accuracy of EEMSS for each user. We compared the state predicted by EEMSS with the ground truth state recorded in the user's diary for every time step. Accuracy is defined as the number of correct predictions over the total number of predictions. Figure 9 shows the state recognition accuracy for the ten participants in this study. The recognition accuracy varies slightly from one user to another simply due to different user behaviors during the experiment. The average recognition accuracy over all users is found to be 92.56% with a standard deviation of 2.53%.

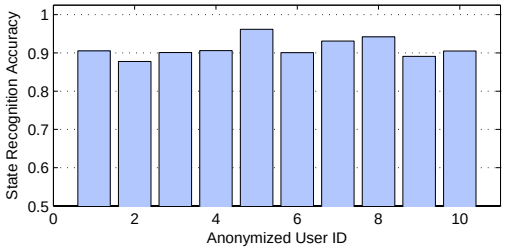

Figure 9: The state recognition accuracy for all 10 participants.

	At some place	Walking	Vehicle
At some place	99.17%	0.78%	0.05%
Walking	12.64%	84.29%	3.07%
Vehicle	10.59%	15.29%	74.12%

Table 7: EEMSS confusion matrix of recognizing "Walking", "Vehicle" and "At some place". The first column represents the ground truth while the first row represents the recognition results. For example, "At some place" is recognized as "Walking" in 0.78% of the time.

We also examine the overall state recognition accuracy in terms of the confusion matrix of identifying "Walking", "Vehicle" and "At some place". We do not present the confusion matrix for all 11 states introduced in Section 4 due to the fact that "Working", "Meeting", and "Office_loud" states can be discriminated based only on audio sensing, whereas in Section 6.4 we already showed that our background sound classification can provide more than 90% accuracy, hence we are able to aggregate them together as "At office". Similarly, "Home_speech", "Home_loud" and "Resting" can be summarized as "At home". Meanwhile, the sets of states described above can be discriminated using only location information. For instance, "Working" and "Meeting" and "Office_loud" are all characterized by their office location while "Resting", "Home_loud" and "Home_speech" all take place at home. In Section 6.2 we already explained that performing WiFi scan can detect location with certainty, hence we are able to treat 9 states including 'Resting", "Home_talking", "Home_entertaining", "Working", "Meeting", "Office_loud", "Place_quiet", "Place_speech" and "Place_loud" as a super state: "At some place" to verify state recognition accuracy. Table 7 shows the corresponding confusion matrix. The first column represents the ground truth while the first row represents the returned states by EEMSS. It can be seen that the accuracy is very high when the user is staying at some place such as home, office, etc. compared to traveling outdoors. From this table, 12.64% of walking time and 10.59% of vehicle time is categorized as "At some place". This is because that GPS readings are unavailable due to the device limitations which causes the location timeout and hence the system considers that the user has entered some place. However, this false conclusion can be self-corrected since the accelerometer is able to monitor the motion of the user when he or she is considered still and hence GPS will be turned back on immediately as the user keeps moving. The reason that riding a vehicle is recognized as walking is due to the fact that although we have implemented algorithms that

	Walking	Vehicle	At some place
Walking	N/A	< 40 sec	< 5 min
Vehicle	< 1.5 min	N/A	N/A
At some place	< 1 min	N/A	N/A

Table 8: Average state transition detection latency.

prevent the system to consider regular slow motions of vehicles as walking, there exists extreme conditions where the vehicle is moving very slowly and thus being recognized as the wrong state. We plan to incorporate other mechanism in EEMSS such as combining more than one sensor readings such as accelerometer and GPS to differentiate one's mode of travel.

Note that the difference between Table 7 and Table 5 is that in Table 5 the motion classification is performed based on accelerometer readings whereas in Table 7 we show the recognition accuracy results of EEMSS system. Specifically, in our current EEMSS system implementation, once the user is moving outdoors, GPS will be continuously sampled which not only provides location update but acts as a single source of mode of travel classification such as walking and riding a vehicle. Accelerometer is not turned on during this period to reduce energy cost. However, when GPS becomes unavailable accelerometer could be activated to monitor and detect one's motion as well as mode of travel.

7.2.3 State Transition Detection Latency

Necessary sensors have been assigned for each user state by the sensor management scheme to monitor state transition, and reading changes of specific sensors indicate that the user status has been updated. Thus, state transition detection latency is mainly bounded by the duty cycles of the monitoring sensors as well as relevant parameters such as GPS location request timeout value and the ones used in classification module for increasing recognition accuracy. The entries in Table 8 illustrate the state transition detection latencies among "Walking", "Vehicle" and "At some place" by the EEMSS. It can be seen that Vehicle mode could be quickly detected since only one or two GPS queries are required to identify the sudden change on user location. The time required to recognize that the user has arrived at some place is less than 5 minutes which is the period allowed for GPS location request timeout, and after which a WiFi scan will be performed immediately to recognize one's current location. Note that it takes longer time to detect the transition from riding a vehicle to walking because such a period is required to distinguish the slow motion of vehicle and walking. Since the accelerometer is sampled every 6 seconds when the user is still, user motion can be detected in less than 6 seconds and the user will be considered walking as soon as WiFi scan is triggered by accelerometer and indicates that the user has moved out of his recent position. Such a process normally takes less than 1 minute, as shown in Table 8. Similarly, the background sound change will be detected in less than 3 minutes which is the duty cycle for microphone (this is not shown in the table).

7.2.4 Device Lifetime Test

The benefit of low energy consumption of EEMSS has been verified by researchers through lab studies lasting for 12 days. During each day of the study period two researchers

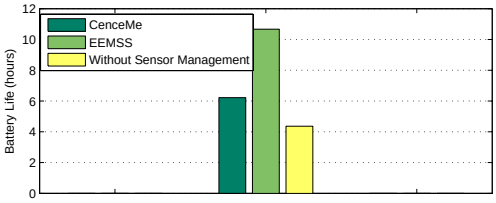

Figure 10: Average N95 lifetime comparison of running EEMSS, CenceMe and the case where all sensors are powered.

have each carried a fully charged N95 smart phone with EEMSS application running on the background and the device lifetime has been examined. The average device lifetime result with EEMSS application running on a fully charged Nokia N95 device is 11.33 hours with regular cell phone functionalities. Note that the device lifetime may vary due to different user behaviors. For example, if the user stays at home most of the time with little activities, the device may stay active for much longer time since only accelerometer and microphone will be periodically monitored as compared to the case where one spends most of the time traveling outdoor and the device lifetime will significantly decrease due to extensive GPS usage.

We have also examined the device lifetime by turning on GPS, accelerometer and microphone on N95 device with the same sampling frequencies as used in EEMSS, and WiFi scanning is performed every 5 minutes (Note that in EEMSS the WiFi scan is only conducted when the user is leaving or arriving at some places). The device lifetime is found to be less than 5 hours regardless of user activity since no sensor management mechanism is implemented and all the sensors will be periodically sampled until running out of battery, shown as "Without Sensor Management" in Figure 10. In [17] it has been shown that the CenceMe application can last for around 6.22 hours with no other applications running on the phone. Although the comparison may not be comprehensive it is the only known result so far that describes the device lifetime with an mobile urban sensing application running. Note that there exist some differences between CenceMe and EEMSS implementations. For example, CenceMe adopts an audio sampling duration of 30 seconds and the implements a 60 seconds duty cycle for the microphone, whereas in EEMSS the audio sampling duration is only 4 seconds and the microphone duty cycle is 180 seconds. On the other hand, CenceMe implements an approximately 10 minutes duty cycle for GPS whereas in EEMSS GPS will be turned on continuously when the user is moving outdoors to provide location tracking. Moreover, CenceMe contains data upload cycles and Bluetooth probing that require extra power usage which are not implemented in our system. The results of battery life durations are summarized in Figure 10, and it can be seen that EEMSS gains more than 75% running hours compared to CenceMe and even larger amount compared to the case where a urban sensing is conducted without sensor management.

To visualize the effect of sensor management, Figure 11 illustrates the energy consumption model of our system at a glance in a 20 minutes interval when the user walks from his office to a library. It can be seen that at the beginning of the

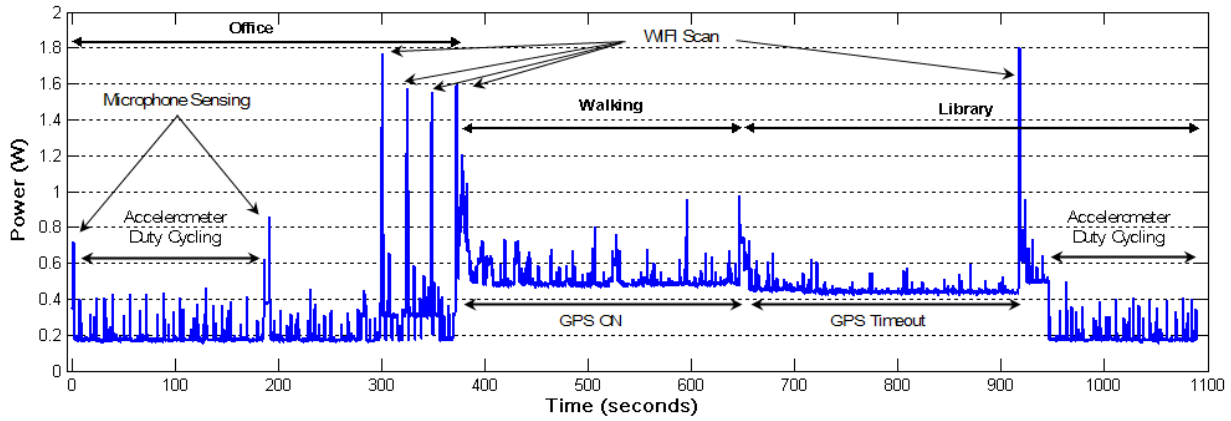

Figure 11: Power usage at a glance.

test when the user is sitting in the office, only accelerometer and microphone is being sampled to detect user movement and identify background sound type. When movement is detected, WiFi scanning will be performed to determine whether the user has left the recent location. Note that multiple WiFi scans may be required until the user leaves his previous position. As the user walks towards the library, GPS is turned on in order to provide positioning information and we allow 5 minutes for GPS timeout as the user enters the library where no GPS signal can be received. Finally, a WiFi scan is performed to recognize the current location and accelerometer will be turned back on for user motion detection.

8. CONCLUSIONS AND FUTURE WORK DIRECTIONS

Mobile device based sensing is able to provide rich contextual information about users and their environment for higher layer applications and services. However, the energy consumption by these sensors, coupled with limited battery capacities, makes it infeasible to be continuously running such sensors.

In this paper, we presented the design, implementation, and evaluation of an Energy Efficient Mobile Sensing System (EEMSS). The core component of EEMSS is a sensor management scheme for mobile devices that operates sensors hierarchically, by selectively turning on the minimum set of sensors to monitor user state and triggers new set of sensors if necessary to achieve state transition detection. Energy consumption can be reduced by shutting down unnecessary sensors at any particular time. Our implementation of EEMSS was on Nokia N95 devices that uses our sensor management scheme to manage built-in sensors on the N95, including GPS, WiFi detector, accelerometer and microphone in order to achieve human daily activity recognition. We also proposed and implemented novel classification algorithms for accelerometer and microphone readings that work in real-time and lead to good performance. Finally, we evaluated EEMSS with 10 users from two universities and were able to provide a high level of accuracy for state recognition, acceptable state transition detection latency, as well as more than 75% gain on device lifetime compared to existing systems.

For future work, we decide to apply machine learning techniques to our system design such that user's specific behavior could be learned to improve the recognition accuracy. How to optimally assign sampling rates to sensors based on different user states will be extensively investigated, and we plan on designing more sophisticated algorithms that dynamically assign sensor duty cycles to further reduce the energy consumption while maintaining low latency for state transition detection as well as high recognition accuracy. We also plan on implementing our sensor management scheme on more complex sensing applications which contain many more types of sensors. Lastly, monitoring user context with privacy concern gives us another promising research direction.

9. REFERENCES

[1] A. T. Campbell, S. B. Eisenman, K. Fodor, N. D. Lane, H. Lu, E. Miluzzo, M. Musolesi, R. A. Peterson, and X. Zheng. Transforming the social networking experience with sensing presence from mobile phones. In *Proceedings of SenSys*, Raleigh, NC, USA, 2008.

[2] B. Hoh, M. Gruteserand R. Herring, J. Ban, D. Work, J. Herrera, A. M. Bayen, M. Annavaram, and Q. Jacobson. Virtual trip lines for distributed privacy-preserving traffic monitoring. In *Proceedings of MobiSys*, Breckenridge, CO, USA, June 2008.

[3] Facebook. http://www.facebook.com.

[4] MySpace. http://www.myspace.com.

[5] H. W. Gellersen, A. Schmidt, and M. Beigl. Multi-sensor context-awareness in mobile devices and smart artifacts. *Mobile Networks and Applications*, 7(5):341–351, 2002.

[6] T. Stiefmeier, D. Roggen, G. Troster, G. Ogris, and P. Lukowicz. Wearable activity tracking in car manufacturing. In *Pervasive Computing*, volume 7, pages 42–50, April-June 2008.

[7] A. Andrew, Y. Anokwa, K. Koscher, J. Lester, and G. Borriello. Context to make you more aware. In *Proceedings of ICDCSW*, Toronto, Ontario, Canada, 2007.

[8] J. Lester, B. Hannaford, and G. Borriello. Are you with me? - using accelerometers to determine if two devices are carried by the same person. In *Pervasive Computing*, pages 33–50, 2004.

[9] T. Choudhury, G. Borriello, S. Consolvo, D. Haehnel, B. Harrison, B. Hemingway, J. Hightower, P. Klasnja, K. Koscher, An. LaMarca, J. A. Landay, L. LeGrand, J. Lester, A. Rahimi, A. Rea, and D. Wyatt. The mobile sensing platform: An embedded activity recognition system. In *Pervasive Computing*, volume 7, pages 32–41, 2008.

[10] N. D. Lane, H. Lu, S. B. Eisenman, and A. T. Campbell. Cooperative techniques supporting sensor-based people -centric inferencing. *Lecture Notes in Computer Science*, 5013/2008:75–92, 2008.

[11] P. Zappi, T. Stiefmeier, E. Farella, D. Roggen, L. Benini, and G. Troster. Activity recognition from on-body sensors by classifier fusion: sensor scalability and robustness. In *Proceedings of ISSNIP*, 2007.

[12] L. Bao and S. S. Intille. Activity recognition from user-annotated acceleration data. *Pervasive Computing*, pages 1–17, 2004.

[13] A. Schmidt, K. A. Aidoo, A. Takaluoma, U. Tuomela, K. V. Laerhoven, and W. V. D. Velde. Advanced interaction in context. In *Proceedings of HUC*, Karlsruhe, Germany, 1999.

[14] J. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M. B. Srivastava. Participatory sensing. In *WSW Workshop at SenSys*, Boulder, Colorado, USA, 2006.

[15] D. Siewiorek, A. Smailagic, J. Furukawa, N. Moraveji, K. Reiger, and J. Shaffer. Sensay: a context-aware mobile phone. In *Proceedings of ISWC*, White Plains, NY, USA, 2003.

[16] E. Miluzzo, N. D. Lane, S. B. Eisenman, and A. T. Campbell. Cenceme - injecting sensing presence into social networking applications. In Gerd Kortuem, Joe Finney, Rodger Lea, and Vasughi Sundramoorthy, editors, *EuroSSC*, volume 4793 of *Lecture Notes in Computer Science*, pages 1–28. Springer, 2007.

[17] E. Miluzzo, N. Lane, K. Fodor, R. Peterson, S. Eisenman, H. Lu, M. Musolesi, X. Zheng, and A. Campbell. Sensing meets mobile social networks: The design, implementation and evaluation of the cenceme application. In *Proceedings of SenSys*, Raleigh, NC, USA, November 2008.

[18] S. Gaonkar, J. Li, R. R. Choudhury, L. Cox, and A. Schmidt. Micro-blog: sharing and querying content through mobile phones and social participation. In *Proceedings of MobiSys*, Breckenridge, Colorado, USA, 2008.

[19] E. Welbourne, J. Lester, A. LaMarca, and G. Borriello. Mobile context inference using low-cost sensors. In *Workshop on Location- and Context-Awareness*, Boulder, Colorado, USA, 2005.

[20] P. Mohan, V. Padmanabhan, and R. Ramjee. Nericell: Rich monitoring of road and traffic conditions using mobile smartphones. In *Proceedings of SenSys*, Raleigh, NC, USA, 2008.

[21] K. Whitesell, B. Kutler, N. Ramanathan, and D. Estrin. Determining indoor air quality from images of an air filter captured on cell phones. In *ImageSense Workshop at SenSys*, Raleigh, NC, USA, 2008.

[22] J. Lester, T. choudhury, G. Borriello, S. Consolvo, J. Landay, K. Everitt, and I. Smith. Sensing and modeling activities to support physical fitness. In *Proceedings of UbiComp*, Tokyo, Japan, 2005.

[23] S. Biswas and M. Quwaider. Body posture identification using hidden markov model with wearable sensor networks. In *BodyNets Workshop*, Tempe, AZ, USA, March 2008.

[24] M. Annavaram, N. Medvidovic, U. Mitra, S. Narayanan, D. Spruijt-Metz, G. Sukhatme, Z. Meng, S. Qiu, R. Kumar, and G. Thatte. Multimodal sensing for pediatric obesity applications. In *UrbanSense08 Workshop at SenSys*, Raleigh, NC, USA, November 2008.

[25] T. Gao, C. Pesto, L. Selavo, Y. Chen, J. Ko, J. Kim, A. Terzis, A. Watt, J. Jeng, B. Chen, K. Lorincz, and M. Welsh. Wireless medical sensor networks in emergency response: Implementation and pilot results. In *HST*, Waltham, MA, USA, May 2008.

[26] D. Jea, J. Liu, T. Schmid, and M. Srivastava. Hassle free fitness monitoring. In *HealthNet Workshop at MobiSys*, Brekenridge, CO, USA, 2008.

[27] W. H. Wu, L. K. Au, B. Jordan, T. Stathopoulos, M. A. Batalin, W. J. Kaiser, A. Vahdatpour, M. Sarrafzadeh, M. Fang, and J. Chodosh. Smartcane system: An assistive device for geriatrics. In *BodyNets Workshop*, Tempe, AZ, USA, 2008.

[28] M. A. Viredaz, L. S. Brakmo, and W. R. Hamburgen. Energy management on handheld devices. *ACM Queue*, 1:44–52, 2003.

[29] A. Krause, M. Ihmiq, E. Rankin, D. Leong, G. Smriti, D. Siewiorek, A. Smailaqic, M. Deisher, and U. Senqupta. Trading off prediction accuracy and power consumption for context-aware wearable computing. In *Proceedings of ISWC*, Osaka, Japan, 2005.

[30] P. Pillai and K. G. Shin. Real-time dynamic voltage scaling for low-power embedded operating systems. In *Proceedings of SOSP*, Banff, Alberta, Canada, 2001.

[31] E. Shih, P. Bahl, and M. J. Sinclair. Wake on wireless: an event driven energy saving strategy for battery operated devices. In *Proceedings of MobiCom*, Atlanta, Georgia, USA, 2002.

[32] J. Sorber, N. Banerjee, M. D. Corner, and S. Rollins. Turducken: hierarchical power management for mobile devices. In *Proceedings of MobiSys*, Seattle, Washington, USA, 2005.

[33] S. Kang, J. Lee, H. Jang, H. Lee, Y. Lee, S. Park, T. Park, and J. Song. Seemon: scalable and energy-efficient context monitoring framework for sensor-rich mobile environments. In *Proceedings of MobiSys*, Breckenridge, CO, USA, 2008.

[34] Nokia Energy Profiler. http://www.forum.nokia.com/main/resources /user_experience/power_management /nokia_energy_profiler/.

[35] SKYHOOK Wireless. http://www.skyhookwireless.com/.

[36] G. Lu and T. Hankinson. A technique towards automatic audio classification and retrieval. In *Proceedings of ICSP*, Beijing, China, 1998.

[37] B. Gajic and K.K. Paliwal. Robust speech recognition in noisy environments based on subband spectral centroid histograms. In *IEEE Transactions on speech and audio processing*, pages 600–608, 2006.