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

[†]Department of Electrical Engineering, University of Southern California, USA
[‡]School of Computer Science, Carnegie Mellon University, 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 is difficult due to battery life limitations. In this paper, we present the framework design for an Energy Efficient Mobile Sensing System (EEMSS) that powers only necessary and energy efficient sensors and manages sensors hierarchically to recognize user state as well as detect state transitions. We also present the design, implementation, and evaluation of EEMSS that automatically recognizes user 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 it increases the smart phone's battery life by more than 75% while maintaining both high accuracy and low latency in identifying transitions between end-user activities.

1. INTRODUCTION

As Moore's law continues doubling the number of transistors in unit area every 18 months, defying several pessimistic predictions, mobile phones are packing more functionalities onto a single chip. A large fraction of the growth in functionality is achieved through integrating complex sensing capabilities on mobile devices. Even mid-range mobile phones now pack features that were primarily in the previous generation's high-end mobile devices. For instance, 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 application side, ambient sensing [1] has become a primary input 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]. Using AI and data mining techniques, a user's context can also be used to learn about the interactions between user's behavior and the surrounding environment. Combining both the technology push and demand pull, more and more context aware applications are trying to utilize various data sensed by existing embedded sensors. However, the critical piece that provides services to these applications is user state recognition. A user state contains a combination of features such as motion, location and background condition that together describe one's current condition. By extracting more meaningful characteristics of devices, 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.

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 completed drained within six hours if the GPS receiver is turned on, whether it can obtain GPS readings or not. Hence, excessive power consumption may become a major obstacle to broader acceptance context-aware mobile applications, no matter how useful the service may be.

To address this problem, 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. Based on various sensor readings, EEMSS automatically recognize user state which describes the user's real-time condition including one's motion (such as running and walking), location (such as staying at home or on a freeway) and background

environment (such as sound and crowd level). 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 system input and is used by our sensor assignment functional block to control sensors based on a user's current condition at any specific time.

The benefits of our sensor management scheme are threefold. First, the state descriptor supports flexibility since the set of user states to be classified can be configured based on the application requirements. Second, to achieve energy efficiency, the sensor management scheme assigns the minimum set of sensors at any particular time as required and invokes new sensors when state transitions happen. Other sensors are activated when this minimum set of sensors detects a state transition, with a corresponding new minimum set of sensors assigned. In this way, a hierarchical sensor management, which reconfigures the sent of active sensors dynamically is achieved in state transitions. Lastly, our sensor management scheme can be potentially extended as a middleware that conducts sensor operations and provides contextual information to higher layer applications or much more sophisticated sensing projects with multiple types of devices and sensors involved.

EEMSS is currently implemented and was evaluated on Nokia N95 devices. EEMSS currently makes use of data from the accelerometer, WiFi detector, GPS, and microphone, which are all built-in sensors on the N95. EEMSS also incorporates novel and efficient classification algorithms for real-time user activity and background sound recognition. Using these algorithms, EEMSS can currently classify such user states as "Walking", "Vehicle", "Resting", "Home _talking", "Home_entertaining", "Working", "Meeting", "Loud _office", "Quiet_place", "Speech_place", and "Loud_place", all specified by a combination of features obtained from different sensor readings. 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%.

The remainder of this paper is organized as follows. In Section 2, we present previous relevant works and their relations to our study. In Section 3, we propose the sensor management scheme which is the core component of real-time energy efficient mobile sensing system design. In Section 4, we introduce a case study of EEMSS on Nokia N95 devices and present its 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 also 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. 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. They developed a jacket with various sensors attached is developed in their project.

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 [7, 8, 9]. For example, more than 20 human activities including walking, watching TV, running, stretching and so on can be recognized with fairly high accuracy [10]. While it is still not clear what kind of classification algorithms works the best, most of the existing works require accelerometer sensor(s) to be installed on pre-identified position(s) near human body.

One way to make the sensing process less obtrusive is to use the off-the-shelf mobile devices such that no external sensors are needed. Several works have been conducted by using the commodity cell phones as platforms [11, 12, 13, 14, 15, 16]. For example, "CenceMe" [13] 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 [14]. Similarly, "Sensay" [12] 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.

Researchers from different fields have studied and implemented a large number of sensors including GPS, Bluetooth, WiFi detector, oxygen sensor, accelerometer, electrocardiograph sensor, temperature sensor, light sensor, microphone, camera, and so on, in projects such as urban/paticipatory sensing, activity recognition and health monitoring [13, 14, 11, 17, 18, 19, 20].

This variety of sensors enables sensing systems to provide extremely rich context information of the users for higher layer applications such as human status tracking, social networking, and location based services. However, limited battery life significantly constrains how long and how often i mobile sensing can be done. The case becomes more severe if the sensors are all turned on continuously without any controlling mechanism.

The problem of power management on mobile devices has been well-explored. Viredaz *et al.* [21] surveyed many fundamental but effective methods for saving power on handheld devices. It has been suggested from the architecture point of view that the system hardware should be designed as a collection of inter-connected building blocks that could function independently to enable independent power man-

agement. Dynamic frequency/voltage scaling [22] should be adopted to reduce power consumption by configuring the processor based on the requirements of the executing applications. In other words, the power-saving scheme should be fully customized for real-time power consumption situation and the specific application requirements. However, these methods are more suitable for lower-level system design rather than application development. Event driven power-saving method is another important method to reduce extra power consumption as well. Shih *et. al.* [23] 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, a hierarchical power management method was used in [24]. 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. We also use a hierarchical approach for managing sensors, and do so in such a way that still maintains accuracy in sensing the user's state. A similar idea was explored by the "SeeMon" system [25], 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 our framework design for energy efficient mobile sensing, energy efficiency is achieved by managing sensors in a hierarchical way based on the user's current state. Note that a state may describe the user's real-time condition which could possibly represent one's motion (such as running and walking), location (such as staying at home or on a freeway) and background environment (such as sound and crowd level).

The core component of EEMSS is a *sensor management scheme* that associates user states with particular sensors and contains both the set of necessary sensors need to be monitored and the sequence of future sensors to be turned on to detect state transition. The sensor management scheme uniquely describes the features of each user state by a particular sensing criteria and state transition will only take into 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. 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.

Sensor assignment is achieved by specifying an XML-format state descriptor as system input that contains all the states to be automatically classified by the sensing system 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.

Figure 1 illustrates the general format of a state descriptor. It can be seen that a user state is defined between the "<State>" 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 sufficient for 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 can represent the hierarchical relationship among sensor in a clear manner. Second, the state descriptors 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.

It is important to note that the system designer has to be familiar with the operations of different sensors such as assigning suitable sampling period and duty cycle to each sensor in order to well maintain the sensing functionality. Classification algorithms that recognize user status based on different sensor readings also need to be pre-trained and implemented as part of the system.

Various sensors makes the user's contextual information available in multiple dimensions, from which a rich set of user states can be inferred. Different users or higher layer applications may only be interested in a subset of states with corresponding sensors that provide specific context information. 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, an important 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 context requests, by which our framework design can be potentially adopted.

Here we also propose a possible extension of our EEMSS design. In our current implementation, sensor management rules are manually configured in the XML state descriptor and sensor sampling intervals and duty cycles are chosen based on extensive experiments (details of duty cycle assignments will be discussed in Section 5). In order to provide an

```
<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.

automated sensor assignment mechanism rather than manually specifying sensing parameters, a sensor information database could be built *a priori* on each mobile device that stores the sensor power consumption statistics, sensor operation methods and so on. As a result, 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 automatically generated or computed based on the existing knowledge, thus human input may not be required anymore.

4. EEMSS DESIGN AND IMPLEMENTATION – A CASE STUDY

4.1 EEMSS: The Overview

As a case study, we have implemented the sensor management scheme as the core component of our implementation of EEMSS (energy efficient mobile sensing system) on Nokia N95 devices. The N95 comes with several built-in sensors, including GPS, WiFi detector, accelerometer, and embedded microphone. The goal of the case study is to conduct a prototype implementation 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 the previous section. Table 1 illustrates the set of user states to be recognized by EEMSS and some of the

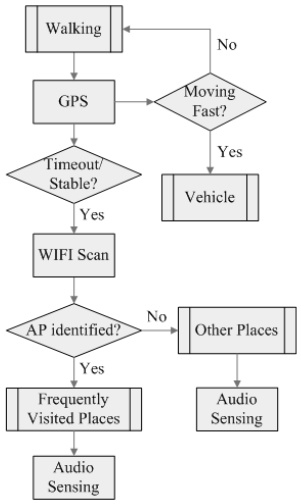

Figure 2: The sequential sensor management rules used to detect state transitions when the user is walking outdoors.

characteristics of these states.

For each user state, the mobile device will follow the sensor management scheme and monitor certain sensors to detect state transition, as shown in Table 1. 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. Recall that the hierarchical sensor management are originally defined in the state descriptor by specifying sensors to be monitored and sensing criteria that trigger state transitions. It can be seen that the only sensor that is being periodically sampled is GPS when the user is walking, which returns both the Geo-coordinates 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-assigned on the device. Finally, the environment can be further sensed based on the audio signal processing result. We will quantify the accuracy and supported device lifetime by our system 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 EEMSS due to either low technology penetration rate or sensitivity to the phone's physical placement. For example, 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), even though a big crowd of people is nearby. The light sensor is also not adopted in our study

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	Lound	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).

because the result of light sensing by mobile phone depends highly on the user's habit of device placement 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 requires manual switching on/off. 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 EEMSS: 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 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 ex-

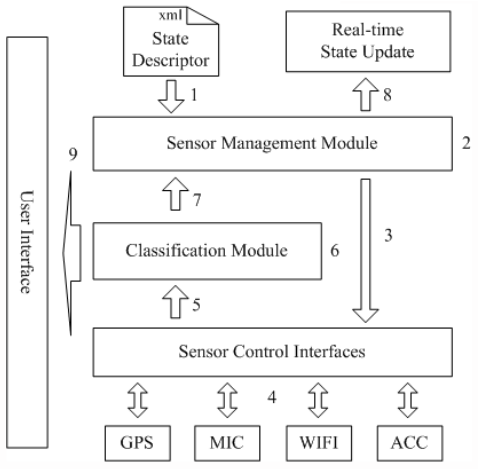

Figure 3: System architecture of EEMSS implementation on Nokia N95. (1)System reads in the XML state descriptor which contains the sensor management scheme. (2)Management module determines the sensors to be monitored based on current user state which is specified by the sensor management scheme. (3)Management 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 consumption summary for different sensors on Nokia N95.

ample, 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 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 the energy consumption of each sensor in the Nokia N95, to understand how to best coordinate them in an effective way. We conducted a series of energy consumption measurements on different built-in sensors, including GPS, WiFi detector, microphone and accelerometer. We also discuss the implementation of duty cycling mechanisms on the sensors to reduce the energy cost.

5.1 Energy Consumption Measurement

The sensors on a mobile phone can be categorized into two classes. The first class includes the accelerometer and microphone. These sensors operate continuously and require an explicit signal to be turned off. Moreover, both the accelerometer and the microphone need to be activated for a period of time to obtain meaningful sensing data. The second class of sensors includes GPS, WiFi detector, and Bluetooth scanner. These sensors gather instantaneous samples, and are automatically turned off when the sampling interval is over.

For both classes, the energy cost of the sensors 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 N95s require a certain amount of time 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. In our work, assisted-GPS is used to help reduce the satellite synchronization time to less than 10 seconds in outdoor locations. 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.

We measure sensor energy consumptions through Nokia Energy Profiler [26], 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 energy consumed by different sensors vary greatly. Among these sensors, accelerometer consumes the least amount of power compared to other sensors. accelerometer is also very sensitive such that it is able to capture the change of body movement which could be an indicator of state transition with high probability. Being sampled periodically, accelerometer could be used 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 and the readings are guaranteed, such as the user is moving outdoors, in order to provide location tracking and mode of travel classification.

5.2 Sensor Duty Cycle Assignment and Computation Time

Although the sensor management scheme guarantees that only the minimum set of sensors are monitored at any specific time, assigning an appropriate duty cycle to each sensor (i.e., the sensor will adopt periodic sensing and sleeping instead of being sampled continuously) should further reduce the energy consumption while keeping the original sensing capabilities.

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 power battery 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 is hence able to be sampled more frequently with smaller 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, 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

Sensor	Duty Cycles	Computation Time
Accelerometer	6 seconds sensing + 10 seconds sleeping	< 0.1 second
Microphone	4 seconds sensing + 180 seconds sleeping	Quiet: < 0.5 second. Loud/Speech: ~ 10 seconds.
GPS	GPS query every 20 seconds, timeout in 5 minutes	< 0.1 second
WiFi scan	Event triggering based sensing	< 0.1 second

Table 3: Sensor duty cycles and device computation time of sensing data.

again. WiFi scanning is event-based rather than something that needs to be done 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 parameters finally adopted by EEMSS (see Table 3) may not be optimal due to the fact that in our current implementation, the parameters are manually tuned based on detection accuracy and energy consumption of each sensor. No optimization or dynamic adjustment has been implemented. We will leave finding the optimal sampling interval and duty cycles, as well as dynamically assigning suitable duty cycles to each sensor based on user's state as future work. Ideally, algorithms can be designed that that could further reduce the energy consumption while maintaining accuracy and low latency.

The device computation time based on sensor duty cycle parameters, including the time for sensor data processing and user status classification, are also summarized in Table 3. Except for loud audio signal processing and classifcation, which takes approximately 10 seconds to complete (mainly consumed at the FFT stage), most of the computations are able to finish immediately, which enables our system to conduct real-time state recognition.

6. SENSOR INFERENCE AND CLASSIFICATION

As discussed in the previous section, the functionalities and features of each sensor have to be systematically studied in order to be better utilized to convey user context information. In this section, we discuss the sensing capabilities and potential human activities that could be inferred from sensors including GPS and WiFi detector. We also propose and implement classification algorithms that are differentiable from previous works which identify user activities as well as background sound types in real-time based on accelerometer and microphone readings.

6.1 GPS Sensing and Mode of Travel Classification

Besides providing real-time location tracking, GPS can be used to detect the user's basic mode of travel since it is able to return both the Geo-coordinates and the moving speed of the user. 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. The mode of travel classifier based on GPS is simply performed by checking the recent moving distance and speed hence we do not present the detail here. The classifier is trained by several location tracking records of user and certain threshold values are identified and implemented into the classification algorithm.

Being periodically queried, GPS on N95 device can also be used to indicate that the user has entered a building or other indoor environment since a location request timeout will occur when the satellite signals are not reachable by the mobile device. [1]

6.2 WiFi Scanning and Usage

The MAC address of visible wireless access points around the user can be returned to the mobile device by performing a WiFi scan. Since MAC address is uniquely assigned, it is possible to tag a particular location by the set of access points visible in that location. Therefore the mobile device is able to automatically identify its current location by simply checking nearby 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 pre-memorized by the device. In our EEMSS implementation, the wireless access points feature of the user's home and office (if applicable) will be pre-recorded for recognition purpose [2]. WiFi scan can also be used to monitor a user's moving range since a wireless access point normally covers an area of radius 20-30m and the previous seen access points will be gone if the user has moving out of that range. In our system implementation, if the user is detected moving continuously by accelerometer, GPS will be turned on to start sampling location information immediately after WiFi scan indicates that the user has left his or her recent range.

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. However, in most of the previous works one or more accelerometer sensors have to be attached to specific body positions. Moreover, several data features have to be extracted from the sensor readings in order to design sophisticated classifiers to recognize detailed activities which are not suitable for real-time implementation.

In our system design, since the mobile phone device is the only source of accelerometer readings which could be placed at various locations due to individual habit, it becomes extremely difficult to perform fully detailed motion classifica-

[1]Obtaining instant speed as well as the location request timeout functionality are both supported by J2ME API.

[2]SKYHOOK wireless [27] system is a good example of implementing wireless access point checking in positioning. However, since the main goal of our work is achieving energy efficiency in mobile sensing, WiFi positioning is implemented only for recognizing one's frequently visited places such as home and office.

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.

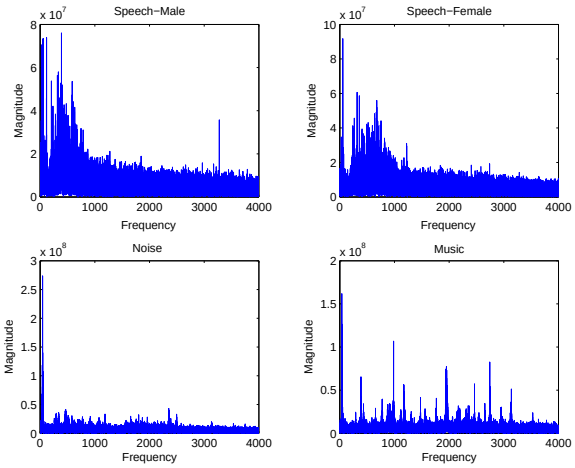

Figure 4: Comparison of frequency domain features of different audio signals.

tion like in previous study [10]. As a matter of fact, only the standard deviation of accelerometer magnitude values is extracted as one of the *independent features* of phone placement in order to conduct real-time motion classification in our work.

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, walking, 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 make sensor control decisions accordingly.

6.4 Real-time Background Sound Recognition

Real time background sound recognition mechanism has also been implemented on Nokia N95 mobile phones. The device records a real time audio clip using microphone sensor and the recorded sound clip will go through two classification steps (Figure 6). 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) [28] (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 4 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 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

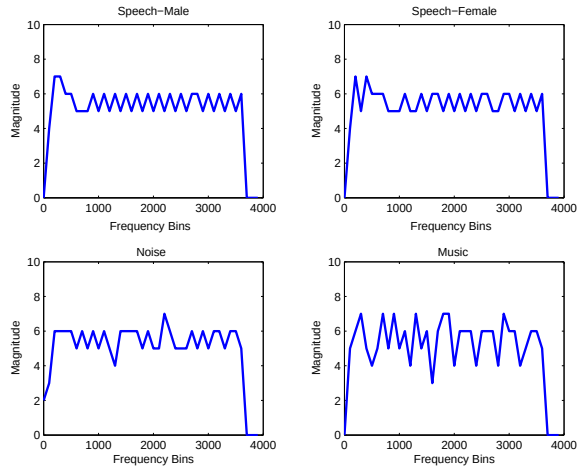

Figure 5: Frequency histogram plots after applying SSCH to sound clips described in Figure 4.

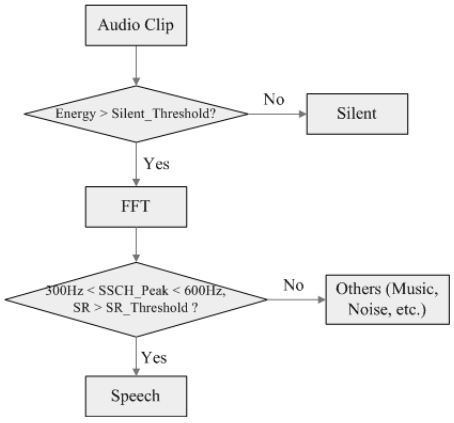

Figure 6: Decision tree based background sound classification algorithm.

accomplish speech detection in real time, we have implemented the SSCH (Subband Spectral Centroid Histogram) algorithm [29] 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 [3] 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 5 illustrates the outputs of applying SSCH algorithm to the sound clips shown in Figure 4. 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 including1085 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 will 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, which is acceptable for real time implementation.

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

	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.

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. Our empirical studies are organized based on the following two phases:

- **Phase I - Lab Study**
  We conducted a lab study over 1.5 months with our team members, to calibrate our classification parameters (e.g., the parameters used to determine mode of travel based on GPS readings), determine the duty cycles for different sensors (e.g: the sampling frequency of accelerometer, GPS and microphone), and perform system energy consumption measurement. During the lab study, participants recorded the ground truth activities and the sensor readings, recognition results were logged by the mobile device for off-line analysis. The energy consumption is measured by using the Nokia Energy Profiler[26] which was turned on in the background continuously to record the power usage.

- **Phase II - User Trial**
  We also conducted a user trial in November 2008 at two different universities. The main purpose of this user trial was to test the EEMSS system in a real setting. We recruited 10 users including undergraduate, graduate students, faculties and family members through online mailing lists and flyers. Each participant was provided with a Nokia N95 device with EEMSS installed. Basic instructions on how to operate the mobile device were also introduced. During this experiment, each user carried an N95 device for two days and

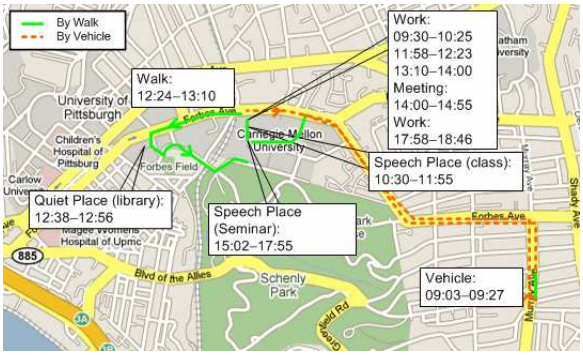

Figure 7: Recognized daily activities of a sample CMU user.

Figure 8: Recognized daily activities of a sample USC user.

kept a diary to manually record ground truth for comparison purpose. A standardized booklet was given to each participant containing three simple questions tagged by time for each record. Specifically, we asked participants to record their motion (e.g.: walking, in vehicle, etc), location, as well as surrounding condition (e.g.: quiet, loud, speech, etc). We then compared diary entries to our recognition results. At the end of the EEMSS evaluation period, we collected more than 260 running hours of EEMSS application, with more than 300 state transitions have been detected by our applications.

7.2 Results

7.2.1 *State Recognition Records*

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. Different types of curves are used to indicate different modes of travel. Specifically, the dashed ones represent vehicle mode whereas the solid ones represent walking mode. Both users have reported that they took buses to school according to their ground truth diaries. The dashed curves are found to match the bus routes perfectly. The solid curves show the traces that the user is walking between home and bus station, within university campus and so on.

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.

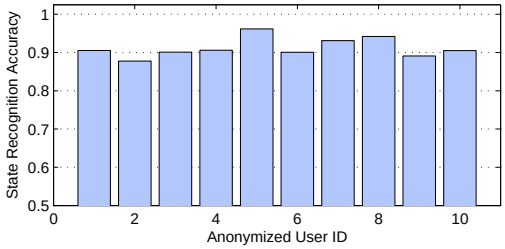

Figure 9: The state recognition accuracy for all 10 participants.

7.2.2 *State Recognition Accuracy*

First, we have examined the overall state recognition accuracy for each participant by comparing the system recognized state log with the ground truth records. The percentage values of detection accuracy is measured by the ratio of number of correctly recognized state records over the total number of records and are shown in Figure 9. According to this figure, the recognition accuracy varies slightly from one user to another, which suggests that our classification algorithms do not bias due to individual varieties. The average recognition accuracy over all users is found to be 92.56% with a standard deviation of 2.53%.

Since in Section 6.4 we have already presented the performance of our background sound classifier which is implemented on EEMSS and the WiFi detection based location recognition provides high accuracy (this is due to the fact that no classification algorithm is implemented here other than performing simple data comparison), we could be able to narrow down our study of recognition accuracy by aggregating all the 11 states into three super states namely "Walking", "Vehicle" and "At some place" featured by one's location and mode of travel information. Table 7 shows the percentage of recognition accuracy for these states. The first column represents the ground truth while the first row represents the returned states of 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 de-

	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.

	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.

vice 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 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

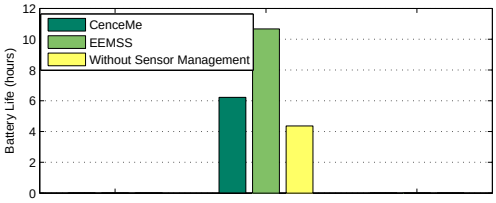

Figure 11: Average N95 lifetime comparison of running EEMSS, CenceMe and the case where all sensors are powered.

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 through lab studies lasting for 12 days. During each day of the study period two researchers 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 tested 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. In [14] 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

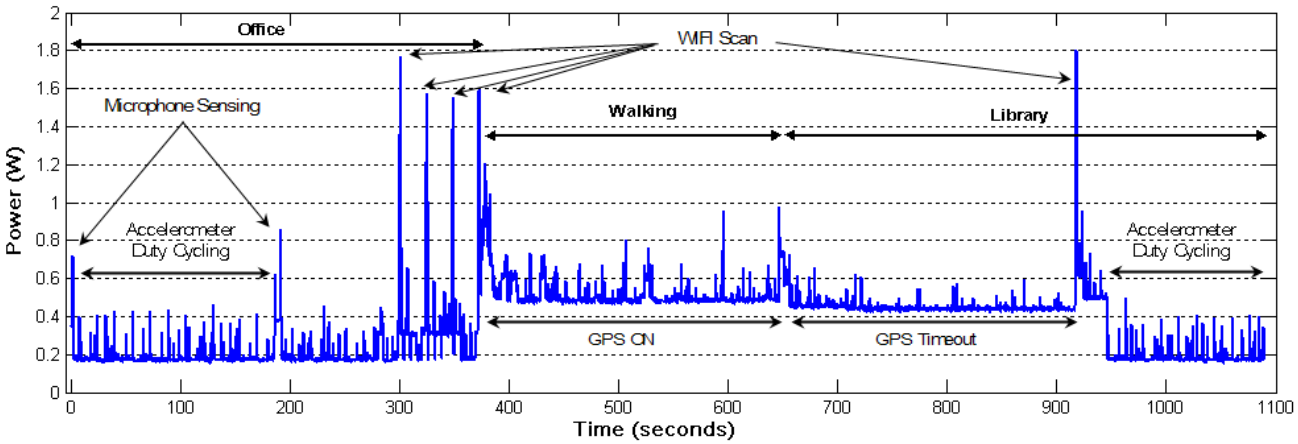

Figure 10: Power usage at a glance.

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 11, 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 visulize the effect of sensor management, Figure 10 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 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. 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 is 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 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.

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 *SenSys '08: Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 367–368, New York, NY, USA, 2008. ACM.

[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 *MobiSys '08:*

Proceeding of the 6th international conference on Mobile systems, applications, and services, pages 15–28, New York, NY, USA, June 2008. ACM.

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

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

[5] Hans W. Gellersen, Albercht Schmidt, and Michael Beigl. Multi-sensor context-awareness in mobile devices and smart artifacts. *Mob. Netw. Appl.*, 7(5):341–351, 2002.

[6] T. Stiefmeier, D. Roggen, G. Troster, G. Ogris, and P. Lukowicz. Wearable activity tracking in car manufacturing. *Pervasive Computing, IEEE*, 7(2):42–50, April-June 2008.

[7] 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. *Pervasive Computing*, 7(2):32–41, 2008.

[8] 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.

[9] 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 *3rd International Conference on Intelligent Sensors, Sensor Networks and Information, (ISSNIP 2007)*, 2007.

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

[11] J. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M. B. Srivastava. Participatory sensing. In *WSW'06 at SenSys '06*, Boulder, Colorado, USA, 2006.

[12] D. Siewiorek, A. Smailagic, J. Furukawa, N. Moraveji, K. Reiger, and J. Shaffer. Sensay: A context-aware mobile phone. pages 248–249. IEEE Computer Society, 2003.

[13] 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.

[14] 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 ACM SenSys 2008*, November 2008.

[15] 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 *MobiSys '08: Proceeding of the 6th international conference on Mobile systems, applications, and services*, pages 174–186, New York, NY, USA, 2008. ACM.

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

[17] 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 *International Workshop on Urban, Community, and Social Applications of Networked Sensing Systems (UrbanSense08)*, November 2008.

[18] S. Biswas and M. Quwaider. Body posutre identification using hidden markov model with wearable sensor networks. In *Proceedings of BodyNets Workshop 2008*, March 2008.

[19] 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 *2008 IEEE International Conference on Technologies for Homeland Security*, May 2008.

[20] J. Lester, T. choudhury, G. Borriello, S. Consolvo, J. Landay, K. Everitt, and I. Smith. Sensing and modeling activities to support physical fitness. In *Ubicomp Workshop*, 2005.

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

[22] P. Pillai and K. G. Shin. Real-time dynamic voltage scaling for low-power embedded operating systems. *SIGOPS Oper. Syst. Rev.*, 35(5):89–102, 2001.

[23] E. Shih, P. Bahl, and M. J. Sinclair. Wake on wireless: an event driven energy saving strategy for battery operated devices. In *MobiCom '02: Proceedings of the 8th annual international conference on Mobile computing and networking*, pages 160–171, New York, NY, USA, 2002. ACM.

[24] J. Sorber, N. Banerjee, M. D. Corner, and S. Rollins. Turducken: hierarchical power management for mobile devices. In *MobiSys '05: Proceedings of the 3rd international conference on Mobile systems, applications, and services*, pages 261–274, New York, NY, USA, 2005. ACM.

[25] 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 *MobiSys*, pages 267–280, 2008.

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

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

[28] G. Lu and T. Hankinson. A technique towards automatic audio classification and retrieval. In *Proceedings of Fourth International Conference on Signal Processing*, pages 1142–1145, Beijing, China, 1998.

[29] 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.