International Journal of Technology (2016) 1: 38-49
ISSN 2086-9614

EXPLORING SIGNIFICANT MOTION SENSOR FOR ENERGY-EFFICIENT CONTINUOUS MOTION AND LOCATION SAMPLING IN MOBILE SENSING APPLICATION

Muhammad Fiqri Muthohar[1*], I Gde Dharma Nugraha[1], Deokjai Choi[1]

[1] *School of Electronics & Computer Engineering, Chonnam National University, Gwangju 61186, South Korea*

(Received: September 2015 / Revised: October 2015 / Accepted: January 2016)

ABSTRACT

The significant motion sensor is a new sensor that promises motion detection at low power consumption. Despite that promise, no known research has explored the usage of this sensor, especially in mobile sensing research. In this study, we explore the utilization of this significant motion sensor for continuous motion and location sampling in a mobile sensing application. A location sensor is known for its expensive power consumption in retrieving the location data, and continuously sampling from it will quickly deplete a smartphone battery. We experiment with two sampling strategies that utilize this significant motion sensor to achieve low power consumption during continuous sampling. One strategy involves utilizing the sensor naively, while the other involves combining with the duty cycle. Both strategies achieve low energy consumption, but the one that combines with the duty cycle achieves lower energy consumption. By utilizing this sensor, mobile sensing research especially that samples data from location or motion sensors, will be able to achieve lower energy consumption.

Keywords: Adaptive sampling; Mobile sensing; Significant motion sensor; Smartphone sensor

1. INTRODUCTION

In today’s world, mobile devices such as a smartphones are so popular that they have become people’s personal computing devices. In some cases, the smartphone is the only computing device that people have, as it is capable of doing computing tasks that people currently need, replacing traditional PCs and even laptops. Smartphones are so personal that they are usually carried around all day by their owners wherever and whenever they go.

Because of this, several studies have been conducted to study human behavior through the smartphones people carry. Data used in these studies is usually collected through a mobile sensing application that runs in the background on the phones. Sensor data that can be collected includes hardware sensor data such as that from an accelerometer and magnetometer or from logical sensors like smartphone radios (cell network, Wi-Fi, Bluetooth); data from a composite sensor such as a step detector can also be collected. We will describe mobile sensing briefly in Subsection 0.

In a mobile sensing application, sensor data is usually collected at predefined intervals. This can lead to missing data if the sensing application is configured sparsely or to battery depletion problem if the interval is too dense. For example, the following issues can arise in location data

*Corresponding author’s email: fiqri.muthohar@gmail.com, Tel. +8262-530-0894, Fax. +8262-530-0894
Permalink/DOI: http://dx.doi.org/10.14716/ijtech.v7i1.1980

collection.

- In the case of a sparse interval, if the interval is configured for the application to collect location data every 15 minutes, it may miss some event. An example might be when the subject stops at a shop along the road during this sensor sleep period.
- If the interval is dense, for example, if the data is collected every 15 or 30 seconds, this will cause a battery depletion problem.

Another issue is data redundancy; when the subject is not moving during a time interval, this can lead to a problem in data storage, as mobile devices usually have limited data space. However, we are only concerned about energy consumption in this study.

A new sensor has been developed to help detect motion that leads to user movement or location change. Although this sensor promises low power consumption to detect motion, currently there is no known research that explores it. This sensor is called a "significant motion sensor," and it will be described in more detail in Subsection 2.2. By utilizing this new sensor, we try to reduce battery consumption in a mobile sensing application that collects location and motion data, particularly in an opportunistic sensing experiment.

Key contributions of our work in this paper can be summarized as follows. This paper, to best of our knowledge, is the first to present the utilization of a significant motion sensor in mobile sensing for continuous motion and location sensor sampling. We present our strategy to utilize this sensor to achieve low power consumption in continuous motion and location sensor sampling. We then develop a mobile application to implement these sampling strategies, and we experiment with our subject to gather the data. Through using the collected data, we evaluate the effectiveness of our strategy in continuous location and motion sampling for a mobile sensing application.

2. BACKGROUND

In this section, we will describe two backgrounds of our work: mobile sensing and the significant motion sensor.

2.1. Mobile Sensing

Mobile sensing is a process that senses users' activity or behavior through their smartphones. In this process, a subject's smartphone reads data from the sensor embedded in it and then does the computation to sense subject behavior or activity. As can be seen, the process is comprised of two steps: mobile sampling and sensing. Sampling is the process of reading raw sensor data from the mobile device. Sensing is the process of inferring the subject's behavior or activities from the sampled sensor data.

This process is not new, as it has occurred since smartphones have had sensors embedded in them. Both researchers and application developers have an interest in this process, as it can improve people's lives as well as product quality. In health-related research, mobile sensing has been conducted to help monitor patients' health condition through their activity or behavior (Raja et al., 2014). In fitness applications, application developers use mobile sensing to show users' workout performance, history, and many other elements. In the security area, researchers are trying to use collected sensor data as a means of implicit authentication, for example, using a gait cycle as a means of implicit authentication in a smartphone (Hoang & Choi, 2014).

Based on user involvement, there are two types of data gathering methods in mobile sensing. Those two data gathering methods are participatory sensing (Burke et al., 2006) and opportunistic sensing (Lane et al., 2010).

In participatory sensing, the subject needs to interact with the sensing application to allow the data collection. In this type of sensing, the sensing application does not need to predict when to do the data collection, because it only collects sensor data when the subjects interact with it. In opportunistic sensing, on the other hand, the person does not need to interact with the sensing application to allow the data collection. The sensing application should determine when it should sense the data from the sensor available on the subject's smartphone. Periodically sensing the data available in the smartphone is one method used for opportunistic sensing.

One of the problems in mobile sensing is the battery life of the smartphone. To continuously gather data from the sensor and store it, a mobile phone consumes a certain amount of energy. After sampling, the data is then processed to determine the subject's activity or behavior, and this computation also consumes power. Current smartphones are capable of doing such computations in a short time; however, the power consumption required by that computation is still a problem. A smartphone's primary function is as a communication device. Therefore, users do not want the mobile sensing application to drain their phone's battery life quickly.

Research on offloading sensing computation to powerful remote computers has already been conducted to reduce battery consumption and to harness the power of cloud computing (Han et al., 2015; Rachuri et al., 2011). In this study, we try to decrease battery consumption in the sampling phase locally through exploring a significant motion sensor, as sending sensor-read data to a remote computer can generate privacy issues.

2.2. Significant Motion Sensor

A significant motion sensor is "a low-power sensor that detects motion that might lead to a change in the user location" (Google Android, 2015b). This sensor is a software-based sensor instead of a hardware-based one. It uses the underlying hardware sensors in the device, such as an accelerometer sensor, as its sources to detect "significant motion" events.

When this "significant motion" is detected, the significant motion sensor triggers an event through the operating system to notify those who are interested in this event. Walking, riding a bike, or sitting in a moving bus are examples of this type of significant motion (Android Open Source Project, 2015). As long as the smartphone is present during those activities, the sensor should periodically trigger or produce the significant motion event. Meanwhile, if the device is in the pocket of a person who is not moving or the device is on a table that is shaking a bit, this activity is not categorized as significant motion (Android Open Source Project, 2015).

This sensor is relatively low-powered compared to other sensors such as accelerometers or gyroscopes, as shown in Table 1.

Table 1 Sensor power requirement example for Samsung Galaxy S4 SHV-300K (battery 3.8 V)

Sensor Name	Power Requirements
Accelerometer sensor	0.25 mA (0.95 mW)
Magnetic field sensor	6.0 mA (22.8 mW)
Gyroscope sensor	6.1 mA (23.18 mW)
Significant motion sensor	0.3 mA (1.14 mW)
Gravity sensor	12.35 mA (46.93 mW)
Linear acceleration sensor	12.35 mA (46.93 mW)

Another benefit of using this sensor is that no manual computation by the application developer is needed, compared to what is needed for other sensors. In another sensor case, for example, an accelerometer sensor, the application needs to wake the CPU to classify whether the user is moving or not, using that sensor. As stated above, the application classifies the sensor data in the CPU, which also consumes power.

The idea of using a low-power sensor is not new, as Ben Abdesslem et al. (2009) also propose this idea. They present power consumption minimization in mobile sensing by using a combination of an accelerometer sensor, GPS, and WiFi.

According to the documentation (Android Open Source Project, 2015), the sensor makes a tradeoff between power consumption and accuracy. The documentation (Android Open Source Project, 2015) states three reasons for this:

1. Saving power was the goal of this sensor.
2. It should avoid a “false positive,” for example, an event triggered when the user is not moving, because this makes the power expensive.
3. A “false negative,” for example, not triggering an event during user movement, is acceptable when it does not occur repeatedly.

Because of this, the significant motion sensor will delay triggering the event. The delays are typically around 2 seconds with a maximum up to 10 seconds, depending on the implementation by the sensor producer. In fact, any application interested in this event will experience a delay, depending on how long it takes for this event broadcast arrives at their listener.

Unfortunately, “fragmentation” is one of the issues of an Android-based smartphone. This “fragmentation” issue occurs because a great many phones using an older version of the operating system (OS) are still available. The capabilities of devices also depend on their vendors (OpenSignal, 2015; Google Android, 2015a). The significant motion sensor is a relatively new sensor. Its availability is limited to recent high-end smartphones and is supported in the recent version of Android OS API level 19, called “KitKat.”[1] However, according to OpenSignal (2015), the number of devices that include the significant motion sensor is increasing each year.

3. SAMPLING STRATEGIES

Although the purpose of the significant motion sensor is to detect whenever movement occurs, it still requires a strategy for sampling the subject location while balancing power consumption and data sampling accuracy. We attempted two strategies for sampling data from smartphones utilizing the significant motion sensor: the adaptive duty cycle and the naïve adaptive sampling. The following sections explain these two sampling strategies.

3.1. Adaptive Duty Cycle

In this strategy, we checked whether the significant motion event was triggered or not, rather than classifying the sensor reading as either an interesting or a not interesting event. This idea was inspired by a learning automata theory (Narendra & Thathachar, 1989) called the *linear reward-inaction learning* scheme. In this scheme, we need to choose an action. While doing that action, this scheme checks whether or not it got the expected result. If it captures the expected result, it increases the probability of doing the next action. If it captures an unexpected result, then it decreases the probability of doing the next action.

Another study (Rachuri et al., 2011) has already explored this scheme. The difference is that the authors needed to classify the sensor reading result first to determine the next cycle ratio. They classified the sensor data reading either in the phone CPU or a remote computer. Classifying the sensor reading in the smartphone CPU is power-expensive. Classifying the sensor readings in a

[1] https://www.android.com/versions/kit-kat-4-4/

remote computer involves overhead in sending the data and involves delays through the network connection, combined with the computation time on the remote computer. In our case, we wanted to sample location and motion sensor data using only the smartphone while maintaining low power consumption.

In our strategy's implementation, we modified the *linear reward-inaction learning* scheme to listen to the significant motion event. We listened to the significant motion event not only during the sampling period but also during the sleep period of the duty cycle. Therefore, our strategy to utilize the significant motion sensor was as follows. If a significant motion event was detected in this duty cycle period, then the next sampling ratio was increased. Otherwise, the next sampling ratio was decreased, if there was no significant motion event detected during the duty cycle period. **Error! Reference source not found.** illustrates the flowchart of this process.

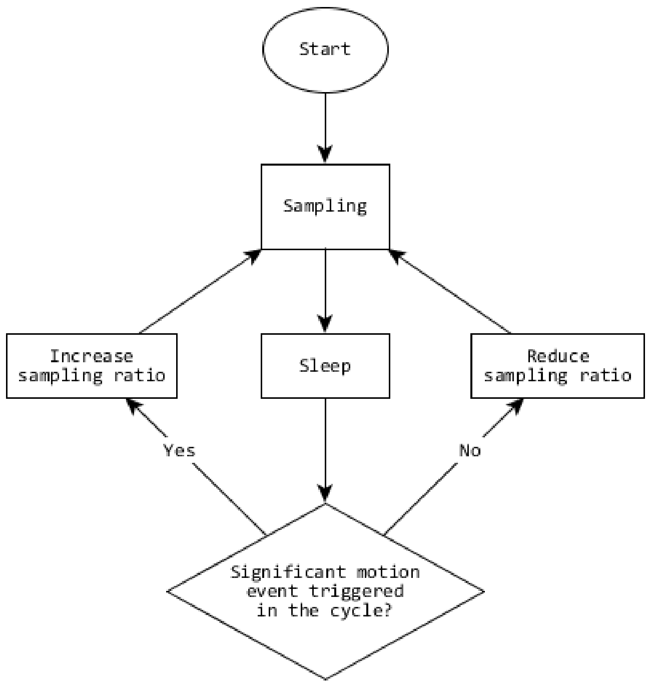

Figure 1 Adaptive duty cycle flowchart

To increase the sampling duty cycle ratio, we used Equation 1. To decrease the sampling duty cycle ratio, we used Equation 2, where T is the calculated sampling duty cycle ratio, $\tilde{T}$ is the previous sampling duty cycle ratio, and α is a constant factor that adjusts the rate to increase or to decrease the sampling ratio.

$$T = \tilde{T} + \alpha(1 - \tilde{T}), \text{where } 0 < \alpha < 1 \tag{1}$$

and

$$T = \tilde{T} - \alpha\tilde{T}, \text{where } 0 < \alpha < 1 \tag{2}$$

3.2. Naïve Adaptive Sampling

This strategy is a naïve usage of the significant motion sensor to start motion and location sensor sampling. We tried this strategy to compare it with our previous strategy, the adaptive duty cycle. In this adaptive sampling strategy, we started directly sampling the sensor data after a significant motion event was triggered instead of using the duty cycle. Then the mobile application listened to the significant motion event to begin the sampling. The sampling session would continue as long as the last significant motion trigger event occurred within the defined time threshold. If the last significant motion event exceeded that time threshold, sampling was stopped, and then the sensor would be put into sleep mode until another significant motion event occurred. **Error! Reference source not found.** illustrates this method using a flowchart.

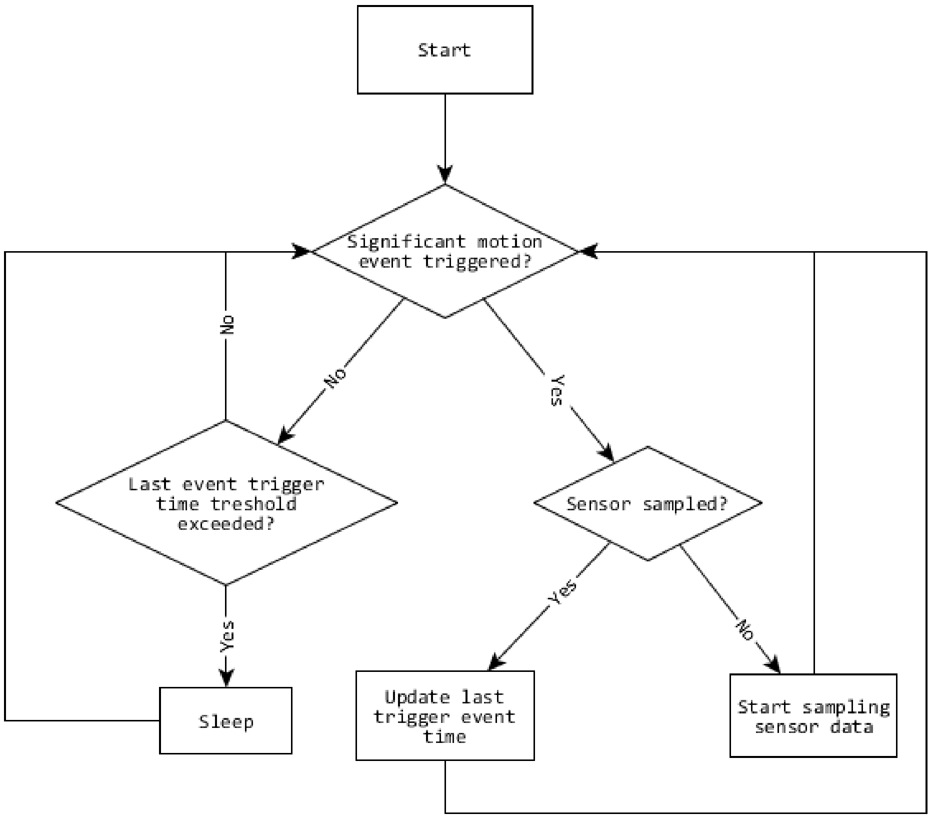
Start
Significant motion event triggered?
No
No
Yes
Last event trigger time treshold exceeded?
Sensor sampled?
Yes
Yes
No
Sleep
Update last trigger event time
Start sampling sensor data

Figure 2 Naïve adaptive sampling flowchart

4. IMPLEMENTATION AND EXPERIMENT SETUP

4.1. Implementation

To implement these strategies, we built a mobile sensing application on top of the "MoST framework"[2] on the Android OS (Cardone et al., 2014). MoST as a modular framework eased the development of the mobile sensing application, as we could add a module that we needed to achieve our goal. This framework was comprised of two major parts: a sensing subsystem and a management subsystem. The sensing subsystem was responsible for the input and output from the sensor (Cardone et al., 2014). Meanwhile, the management subsystem handled the sensing process from the system event, sensing requests and handling power management (Cardone et al., 2014).

For this mobile sensing application, we developed a module that communicated with the power management system of the framework. We implemented both strategies described in Section 0 as a power management input policy. Our module would communicate with the selected sampling strategy whenever it received a significant motion event trigger. Figure 1 displays our module implementation in the MoST architecture.

[2] The MoST framework source code can be checked at: https://github.com/participactunibo/MoST

To change strategies used in the sampling, we provided an option that could select one of these sampling strategies. We chose this approach rather than implementing each strategy in two different apps, to minimize the probability of the subject incidentally running both applications at the same time.

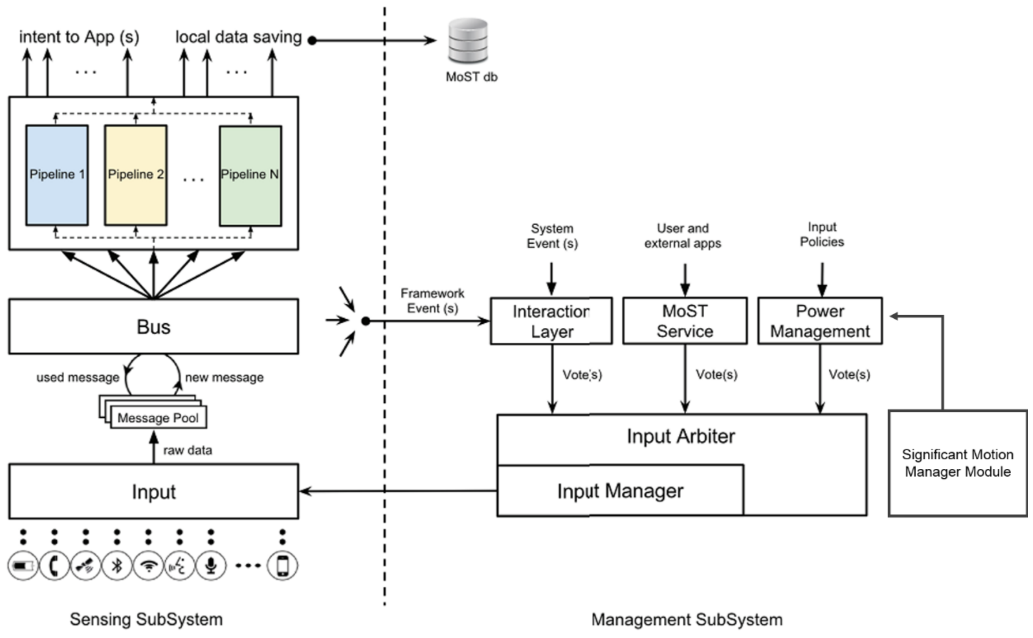

Figure 1 MoST architecture with the significant motion manager module

The α value, used by the adaptive duty cycle strategy in Equations 1 and 2, was set at 0.5 in both equations. The reason we chose this value was to balance the power consumption and the amount of data gathered based on our trial. Rachuri et al. (2011) also attempted varieties of the α value. In their paper, they showed that the α value of 0.5 balanced the power consumption with the accuracy of the collected data (Rachuri et al., 2011). The duty cycle period for the accelerometer was 20 seconds, and the accelerometer sampled continuously during the sampling period. The duty cycle period for the location sensor was 60 seconds, and the location sensor sampled every 30 seconds during the sampling period.

We set the threshold value in the adaptive sampling strategy to 20 seconds. This value was twice the time of maximum delay for the significant motion event to occur. We chose this value because it took time for the application to receive the event signal after the signal was sent from the sensor to the event listener. The application sampled the accelerometer sensor continuously within the sampling threshold time. Moreover, the location sensor was sampled every 30 seconds within the sampling threshold time.

In both strategies, the accelerometer was sampled at the fastest sampling frequency using the *SENSOR_DELAY_FASTEST* (Google Android, 2015b) configuration in the Android API. By using this configuration, the application sampled the accelerometer sensor with a 0-microsecond delay based on the Google Android documentation (2015b). For the location sensor, the application sampled it using the *ACCESS_FINE_LOCATION* permission to be able to get location data from the network provider and GPS provider. The application needs that permission, because sometimes the GPS provider is unavailable, which makes the application unable to retrieve location data.

4.2. Experiment Setup

We installed our mobile sensing application on two kinds of a smartphone with significant motion sensors, the Samsung Galaxy S4[3] and the LG G2[4]. Both phones had the latest Android operating system version, "Android Lollipop,"[5] at the time we used them as data collection devices. Both phones used off-the-shelf configuration, so there was no modification to the hardware configuration or the operating system compared to similar store-bought phone models.

The subjects for this experiment were three of our lab members that each of them had one of those phone models. Our subjects installed the data collection application on their smartphones. Then they opened the application to activate the service for the first time, only because Android does not allow a service to run automatically after being installed. Additionally, we instructed our subjects to take their smartphones whenever they moved.

The subjects then performed their usual activities while the data collection application ran in the background. The subjects' daily activities involved going to the lab, taking classes, going home (or to a dormitory), seeking food, or taking a break. Those daily activities involved changes in the users' locations and movement. However, our subjects mostly stayed in the lab during work hours (9:00 a.m. to 6:00 p.m.). Table 2 shows the daily activity schedule in the experiment scenario. However, this schedule only shows several actions or events in each hour rather than showing the subjects' action during the hour in detail.

Table 2 Experiment scenario daily activity schedule

Time	Monday	Tuesday	Wednesday	Thursday	Friday
08:00	Go to the lab	Go to the lab	Go to the lab	Go to the lab	Go to the lab
09:00	In the lab	In the lab	In the lab	Attend class	In the lab
10:00	In the lab	In the lab	In the lab	Attend class	In the lab
11:00	In the lab	In the lab	In the lab	Attend class	In the lab
12:00	Break	Break	Break	Break	Break
13:00	In the lab	In the lab	In the lab	In the lab	Extra activity
14:00	In the lab	In the lab	In the lab	In the lab	Extra activity
15:00	In the lab	Lab meeting	Attend class	Attend class	In the lab
16:00	In the lab	Lab meeting	Attend class	Attend class	In the lab
17:00	In the lab	In the lab	Attend class	Attend class	In the lab
18:00	Going home	Going home	Going home	Going home	Going home

The application gathered running (foreground) application data, accelerometer sensor data, location data, significant motion sensor events, and battery information data to mimic a real mobile-sensing application. Also, each day, except for weekends, the subjects submitted their data to us and had their system battery log data collected. Data was collected for two weeks, and each week, the subjects collect data using a different sampling strategy. We used this collected data to evaluate both sampling strategies.

5. RESULTS AND DISCUSSION

We collected the battery information using the data collection application, as described in section 0. However, we could not know how much the application had consumed the battery

[3]Specifications can be checked at: http://www.samsung.com/global/microsite/galaxys4/. We used the South Korea version of the phone.
[4]Specifications can be checked at: http://www.lg.com/global/g2. We used the South Korea version of the phone.
[5]h ttps://www.android.com/versions/lollipop-5-0/

energy by using this data. The captured data only contained the battery percentage, timestamp, charging status, and some battery information. This data could project the phone battery drain but could not differentiate the per-application battery energy consumption.

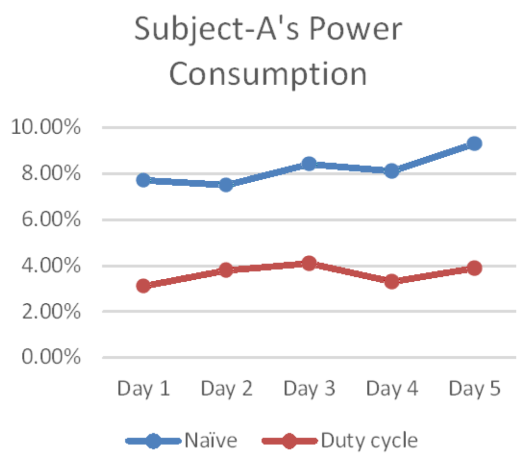

Figure 2 Subject-A's power consumption

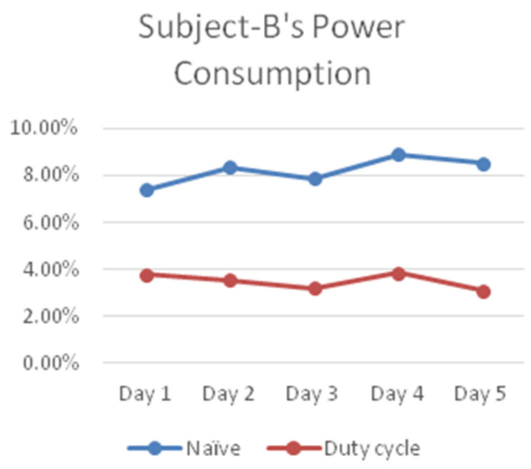

Figure 3 Subject-B's power consumption

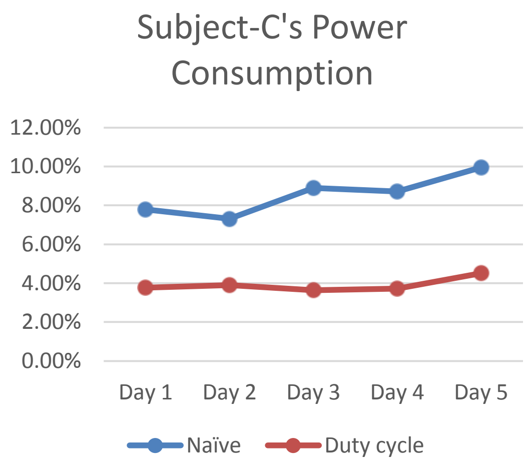

Figure 4 Subject-C's power consumption

We measured the battery consumption in each phone by using the “Battery Historian”[6] tools provided by Google to see how much an application draws power from an Android phone’s battery. The input for this tool was the operating system’s battery log that we collected from each subject every day. By using this tool, we could discern the battery usage from one application to the other. The result from the Battery Historian tool reading included the percentage of battery usage.

Figure 2, Figure 3, and Figure 4 show the results from both sampling strategies from subjects A, B, and C, respectively. In these figures, we can see that using the duty cycle sampling strategy consumed less energy than the naïve sampling strategy.

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x \qquad (3)$$

We used equation (3) to calculate the average of battery consumption. The results were that the duty cycle sampling strategy consumed 3.69 percent of battery life on average while the naïve

[6] The Battery Historian source code can be checked at: https://github.com/google/battery-historian

sampling strategy consumed 8.32 percent of battery life on average. In contrast, collecting sensor data without using any power management policy consumed around 11.5 to 14 percent of battery life over just four hours.

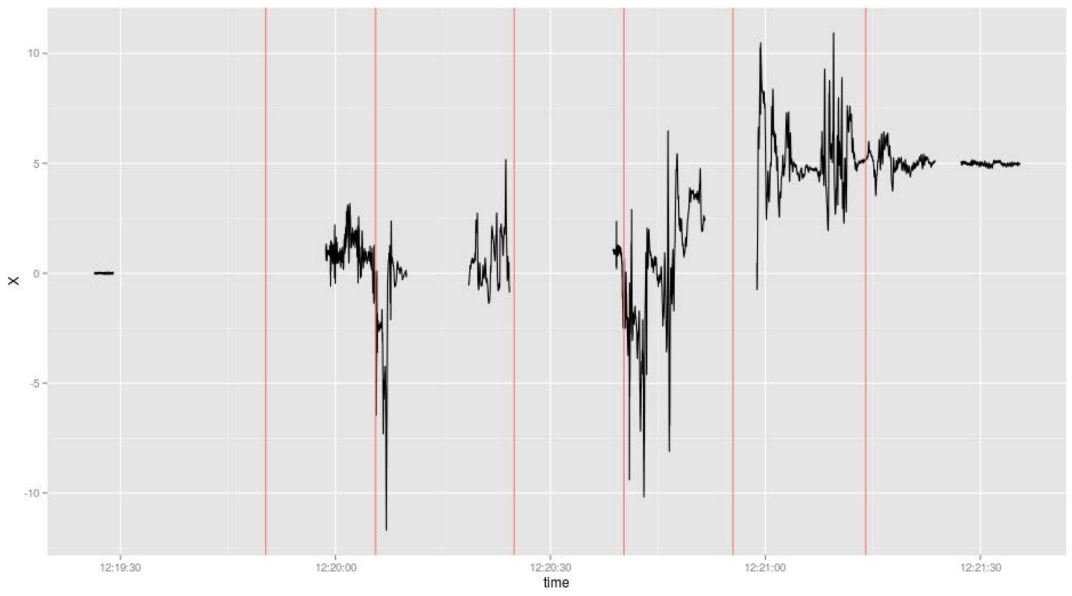

Figure 5 Accelerometer sensor data captured using adaptive duty cycle strategy (red line is the significant motion event)

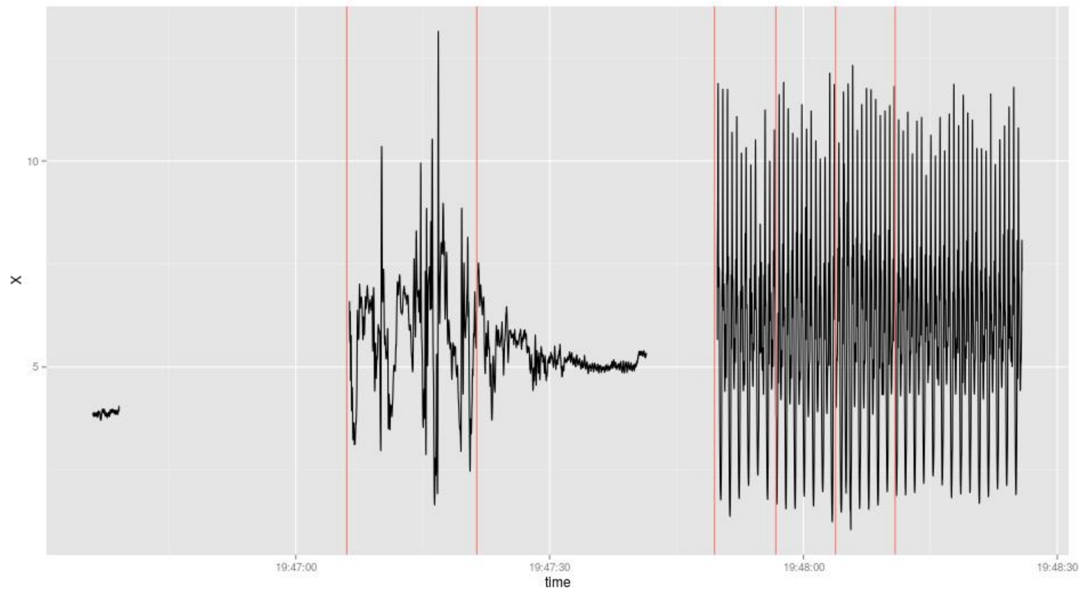

Figure 6 Accelerometer sensor data captured using naïve adaptive sampling strategy (red line is the significant motion event)

These results differ because in the adaptive duty cycle, sampling was done step-by-step rather than using all cycles to do the sampling, as illustrated in Figure 5. Meanwhile, the naïve adaptive sampling method sampled the sensor until the threshold time was exceeded, as illustrated in Figure 6. In both figures, the significant motion event is noted using a vertical red line. Also, the battery drain varied depending on how active the person was. As mentioned in section 0, our subjects were all students who were quite limited in their mobility. In this experiment, all students participated in similar activities during both weeks of the data collection period. Therefore, the battery consumption result does not differ too much among the subjects.

6. RELATED WORK

In our previous work (Mafrur et al., 2015), we already conducted data collection. We used that data for our activity recognition and human behavior research. However, in that study, we were only concerned about continuously collecting data, and we had to make a trade-off between the amount of data that we could collect and the power consumption and storage usage. However, some of the sensor data became meaningless after we processed it because the quantity of the data was insufficient. We hoped to address that trade-off with this study.

7. CONCLUSION AND FUTURE WORK

In this paper, we presented our implementation of energy-efficient continuous motion and location sensor sampling in mobile sensing by strategically utilizing a significant motion sensor to help reduce energy consumption. The mobile application we developed continuously captured motion and location data while maintaining low power consumption during the continuous sampling.

The battery consumption while using the naïve adaptive sampling strategy, which solely utilized the significant motion sensor, was lower than consumption without the adaptive sampling strategy. However, combining the adaptive duty cycle with the significant motion sensor provided a better result, in terms of power consumption, than the naïve adaptive sampling strategy. The adaptive duty cycle strategy consumed 3.69% of the battery, on average, while the adaptive sampling strategy consumed 8.32% of the battery, on average.

For our future work, we want to test this implementation outside our lab with different kinds of occupation. We also want to use this application in our activity recognition and human behavior research.

8. REFERENCES

Android Open Source Project, 2015. *Sensor Types*, Available at: https://source.android.com/devices/sensors/sensor-types.html#significant_motion. Accessed on August 8, 2015

Ben Abdesslem, F., Phillips, A., Henderson, T., 2009. Less is More: Energy-efficient Mobile Sensing with Senseless. In: *the Proceedings of the 1st ACM Workshop on Networking, Systems, and Applications for Mobile Handhelds,* Barcelona, Spain: ACM, pp. 61–62

Burke, J., Estrin, D., Hansen, M., Parker, A., Ramanathan, N., Reddy, S., Srivastava, M.B., 2006. Participatory Sensing. In: *the Proceedings of First Workshop on World-Sensor-Web: Mobile Device Centric Sensory Networks and Applications*

Cardone, G., Cirri, A., Corradi, A., Foschini, L., Montanari, R., 2014. Activity Recognition for Smart City Scenarios: Google Play Services vs. MoST Facilities. *IEEE Symposium on Computers and Communication (ISCC),* Funchal: IEEE, pp. 1–6

Google Android, 2015a. *Android Dashboard,* Available at: http://developer.android.com/about/dashboards/index.html, Accessed on August 6, 2015

Google Android, 2015b. *Sensors Overview,* Available at: http://developer.android.com/guide/topics/sensors/sensors_overview.html, Accessed on August 6, 2015

Han, Q., Liang, S., Zhang, H., 2015. Mobile Cloud Sensing, Big Data, and 5G Networks Make an Intelligent and Smart World. *IEEE Network* (March–April 2015), pp. 40–45

Hoang, T., Choi, D., 2014. Secure and Privacy Enhanced Gait Authentication on Smart Phone. *The Scientific World Journal*, Volume 8, pp. 1–9

Lane, N.D., Miluzzo, E., Lu, H., Peebles, D., Choudhury, T., Campbell, A.T., 2010. A Survey of Mobile Phone Sensing. *Communications Magazine, IEEE* (Sept. 2010), pp. 140–150

Mafrur, R., Nugraha, I.G.D., Choi, D., 2015. Modeling and Discovering Human Behavior from Smartphone Sensing Life-log Data for Identification Purpose. *Human-centric Computing and Information Sciences,* Volume 5, pp. 1–18

Narendra, K.S., Thathachar, M.A., 1989. *Learning Automata: An Introduction*. Prentice-Hall, Inc

OpenSignal, 2015. *Android Fragmentation Visualized,* Available at: http://opensignal.com/reports/2015/08/android-fragmentation/, Accessed on August 8, 2015

Rachuri, K.K., Mascolo, C., Musolesi, M., Rentfrow, P.J., 2011. Sociable Sense: Exploring the Trade-offs of Adaptive Sampling and Computation Offloading for Social Sensing. In: *the Proceedings of the 17th Annual International Conference on Mobile Computing and Networking, Mobicom '11,* pp. 73–84

Raja, A., Tridane, A., Gaffar, A., Lindquist, T., Pribadi, K., 2014. Android and ODK Based Data Collection Framework to Aid in Epidemiological Analysis. *Online Journal of Public Health Informatics*, Volume 5(3), pp. 1–27