Journal of The Korea Society of Computer and Information
Vol. 22 No. 2, pp. 9-19, February 2017
www.ksci.re.kr
https://doi.org/10.9708/jksci.2017.22.02.009

Energy Use Coordinator for Multiple Personal Sensor Devices

Yunseok Rhee*

Abstract

Useful continuous sensing applications are increasingly emerging as a new class of mobile applications. Meanwhile, open, multi-use sensor devices are newly adopted beyond smartphones, and provide huge opportunities to expand potential application categories. In this upcoming environment, uncoordinated use of sensor devices would cause severe imbalance in power consumption of devices, and thus result in early shutdown of some sensing applications depending on power-hungry devices. In this paper, we propose EnergyCordy, a novel inter-device energy use coordination system; with a system-wide holistic view, it coordinates the energy use of concurrent sensing applications over multiple sensor devices. As its key approach, we propose a relaxed sensor association; it decouples the energy use of an application from specific sensor devices leveraging multiple context inference alternatives, allowing flexible energy coordination at runtime. We demonstrated the effectiveness of EnergyCordy by developing multiple example applications over custom-designed wearable senor devices. We show that EnergyCordy effectively coordinates the power usage of concurrent sensing applications over multiple devices and prevent undesired early shutdown of applications.

▸ Keyword : pervasive computing, energy use, mobile sensors

I. Introduction

Continuous sensing applications have been increasingly emerging as a new class of mobile applications [1,2,3]. Moreover, diverse personal sensing devices are broadening application domains [4,5,6,7]; for example a wristband-type sensor allows everyday activity tracking and an earphone-type sensor is enabling daily monitoring of heart rates. A well-known challenge for these applications is to enable continuous sensing and processing with limited battery power - usually until the next expected recharging time [8,9].

Power management problem is becoming more complicated as personal sensor devices are evolving toward open, multi-use architecture; whereas many current commercial devices have closed architecture allowing only a few proprietary applications. Early effort has been made to provide such generic devices with programmable APIs, e.g., Angel [10] and Sensordrone [11]. In the near future, a user will carry multiple such sensors forming a distributed personal computing environment.

In this upcoming multi-sensor environment, a new

• First Author: Yunseok Rhee, Corresponding Author: Yunseok Rhee
*Yunseok Rhee(rheeys@hufs.ac.kr), Division of Computer and Electronic Systems Engineering, Hankuk Univ. of Foreign Studies
• Received: 2017. 02. 08, Revised: 2017. 02. 17, Accepted: 2017. 02. 23.
• This work was supported by Hankuk University of Foreign Studies Research Fund of 2016.

challenge in power management will arise due to inter-dependency of sensor use among concurrent sensing applications. Such interdependency is mainly cuased by collaborative and collective use of sensors, e.g., offloading a sensing workload to another device or fusing data from multiple devices. The dependency becomes more complicated as concurrent applications utilize different sets of devices, which could partially overlap with each other. Given such inter-dependency, uncoordinated uses of devices cause unnecessary contention and early energy dissipation of some sensor devices from the perspective of a whole system. For example, applications that can employ another energy-abundant device might contend for a power-draining sensor with other applications. The unnecessary contention limits availability of other applications that cannot help using that sensor, harming overall system capacity. It is almost impossible for existing solutions limited to a device or an application to handle the inter-dependency among multiple devices.

In this paper, we propose a novel energy use coordination system, EnergyCordy, for multiple open-architecture sensor devices comprising a personal sensing environment. For closed- architecture sensor environments, a usual power management approach has been intra-device optimization [12,13]. In this approach, the association between applications and sensor devices

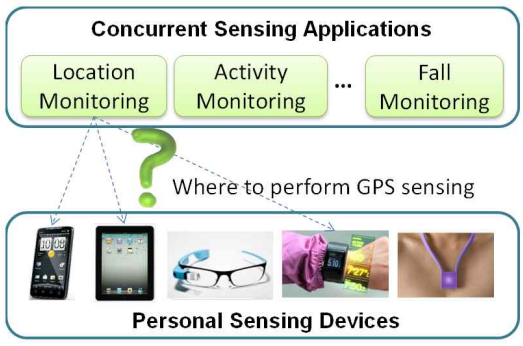

Fig. 1. A holistic view on personal sensor devices.

is predetermined by developers and remains unchanged throughout the application lifetime. The approach mainly focuses on reducing applications' battery use, for example, by adjusting sensing rate or trading off application's functional fidelity. However, an optimized solution for a dedicated device and an application is highly limited in the open, multi-use sensor environment. It is difficult to have overall comprehensive view of energy distribution and available sensor devices as well as other concurrent applications at runtime. Accordingly, they cannot exploit opportunities to resolve contention on specific sensors and prolong the lifetime of multiple applications.

To address the challenges, EnergyCordy coordinates the energy use of concurrent applications with a holistic view of multiple sensor devices and their battery use, instead of managing powers of individual devices separately (Fig. 1). EnergyCordy carefully dispenses limited battery power of distributed sensor devices to competing applications in order to meet the user-preferred goal for application operation. The system-level support of EnergyCordy is essential as it is hardly possible for individual applications to coordinate their energy use with others in an application-level. Also, EnergyCordy completely frees application developers from complicated low-level energy management such as energy availability monitoring and demand profiling.

As a key approach to inter-device power coordination, we propose a relaxed sensor association approach. It delays binding between sensors and applications until runtime when a system has a clear picture of available devices and their remaining battery, as well as running applictions and their power use requirements. This provides much flexibility to coordinate applications' battery use, especially for a device that has little remaining power and many competing applications. A key prerequisite of this approach is to prepare diverse alternative sensor mappings for each sensing application. We observe that many sensing applications have potentials to utilize alternative methods for sensing and context processing; simply from using redundant sensor data sources, to using different combinations of devices placed on different body positions, and even to different sensing modalities with corresponding recognition method [14,15,16].

II. Motivation

2.1 Prototype Sensors

Prototype sensors: In order to reflect a prospective environment of open, multi-use sensing platform, we have designed four sensor devices to incorporate diverse sensing modalities with general programmability support (See Table 1 and Fig. 2). Each device basically has

inertial sensing modules, i.e., 3-axial gyroscope and 3-axial accelerometer. Also, we integrated additional sensing modules matching to each wearing position, e.g., SpO2 sensor on the watch, and ECG sensor on the chest sensor. We have packed the sensors into diverse wearable forms and placed them on various body parts to enable context monitoring, e.g., a right thigh, a chest, a waist, and a left wrist. Each sensor device is powered by a rechargeable Li-Polymer battery with 250mAh capacity. Note that we tried to utilize custom-designed devices as most of the commercially available devices are either not programmable or limited in their sensing modalities. For instance, Android-based smart watches usually support a few sensing modalities such as accelerometer.

Prototype applications: We develop four applications over the above devices: physical activity monitor, heartbeat monitor, environment monitor, and fall detector. Note that the applications are developed to demonstrate the problem addressed by EnergyCordy and its effectiveness, not to claim the novelty of the applications.

Table 1. Custom-designed personal sensor device prototype: wearing position and specification

Sensor ID	Position	Sensing Module	Processing, storage communication module
d_0	Thigh	3-axis accel, 2-axis gyro	- Atmega128 (128KB Instruction EEPROM, 4KB Data EEPROM), - CC2420 (802.15.4, 2.4GHz, 256Kbps) - 1Mb flash memory - Li-Polymer 250mAh
d_1	Chest	3-axis accel, 2-axis gyro, ECG	
d_2	Waist	3-axis accel, 2-axis gyro, Weather (temperature, pressure, illumination)	
d_3	Left Wrist	3-axis accel, 2-axis gyro, SpO_2	

Physical activity monitor is an application to monitor a user's physical activities such as walking, sitting, and bicycling by using wearable sensors. The activity monitoring is important for many context-aware applications to adapt their behavior to the current activity of the user. Also, it can be used to track steps, distance, and calories burned for people who are interested in fitness. There has been much research for physical activity monitoring using on-body accelerometers [15].

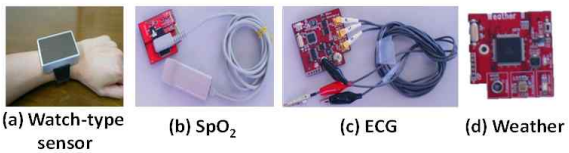

Fig. 2. Hardware and sensing modules of the personal sensor device prototype.

Heartbeat monitor helps users keep their heart rate in a proper level, for exercise efficiency or healthcare. For example, optimal ranges are different according to exercise goals, such as fat-burn or cardiovascular training.

Environment monitor uses sensors to monitor temperature, humidity, dust- or CO2-levels, to warn against the air pollution and weather conditions according to the user's health status. It also enables real-time cartography of the city's air quality and weather maps [16].

Fall detector employs wearable sensors to detect falling. Falling is a frequent and risky event for elderly people. One third of adults aged 65 years or older fall each year, causing serious physiological injury [17]. Wearable sensors with accelerometer or gyroscope enable to monitor the occurrence of a fall in real time and anywhere, calling timely medical attention.

2.2 Motivating Cases

Under the above-described prototype environment, we present an example scenario that describes a potential problem of conflicts in energy use and how EnergyCordy resolves it through coordination. Note that the energy consumptions of the sensor devices in the case are based on actual measurements with our prototypes.

A man in his sixties, living alone, wears a u-watch (d3) equipped with an accelerometer and a SpO2 sensor with remaining energy of 3300J. He also wears a u-shirt(d1) in which an ECG sensor is embedded, currently with 1800J of energy. Every day, he uses a heartbeat monitor application to check his heart rate for a precaution against abnormal condition. This application runs with the SpO2 sensor in the u-watch, expecting longer running time since its remaining energy is larger than the u-shirt. It consumes energy at the rate of 663J/h and accordingly the expected lifetime is 5.0 hours.

He finishes his breakfast and goes to his farm to work. As he leaves his home, a fall detector application is initiated to promptly cope with an accidental fall. This application uses the accelerometer sensor in the u-watch and consumes additional 142 J/hour. Although he expects that applications will keep running until he comes back home, the concurrent use of the applications on the u-watch incurs the quick drain of energy. In consequence, the energy of u-watch is depleted in a short running time of 4.1 hours (See Fig. 3(a)). On depletion of u-watch, the

heartbeat monitor changes its sensing source into the u-shirt instead, which still retains 1243J of energy after 4.1 hours of standby. Now the heartbeat monitor continues by 4.3 additional hours (See Fig. 3(b)). However, the fall detector is already disabled before he comes back home.

EnergyCordy identifies and resolves this problem as shown in Fig. 3(c). It allocates the u-watch solely for the fall detector, and runs the heartbeat monitor with the u-shirt. After serving heartbeat monitor for 6.3 hours, the u-shirt's energy is depleted. At this moment, EnergyCordy rearranges the heartbeat monitor to use the u-watch instead, which has the remaining energy of 1982J to serve the fall detector for 6.3 hours. From now on, the u-watch begins serving both applications, lasting for 2.4 hours. As a result, both applications are able to run for 8.7 hours each and end up at the same time. Not to mention that Souneil could come back home without concerns about the discontinuity of application running.

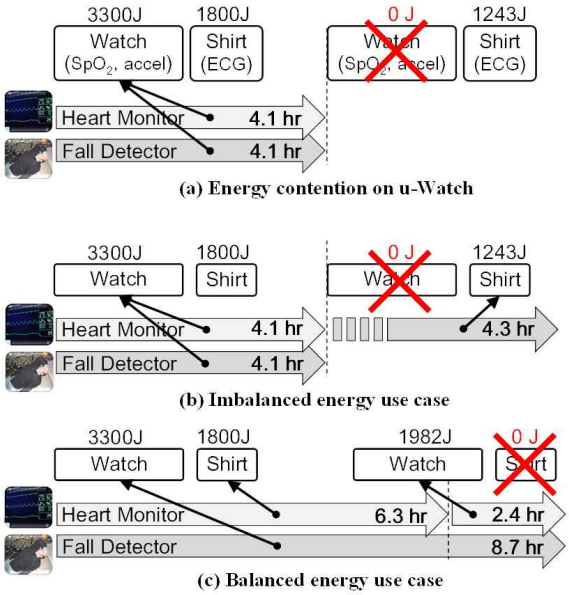

Fig. 3. Energy use coordination: a simple example.

III. System Design

3.1 Architecture Overview

Fig. 4 shows the overall architecture of EnergyCordy and the system environment. We design EnergyCordy as a smartphone-centric architecture where multiple sensor devices are connected to a smartphone in a star topology. The smartphone plays a key role as a sensor manager, which makes a holistic decision for energy

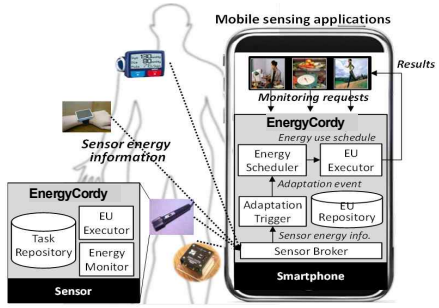

Fig. 4. Architecture overview for Energy use coordination

coordination of sensors. Sensor devices execute sensing and processing tasks as decided by the smartphone and also report runtime energy status required for coordination. Such centralized architecture facilitates efficient system-wide coordination with a global view. Also, a smartphone is a proper device to make the decision since it is always available and has relatively powerful computing and battery resources.

On a smartphone, EnergyCordy runs as a middleware between sensing applications and a mobile OS. The applications specify the type of contexts (e.g., location) of their interest to EnergyCordy, along with the required accuracy. The energy scheduler finds alternative energy use options (EUs) for each application and determines the schedule, EUSched. The adaptation trigger continuously monitors available sensors and their energy status for runtime resource adaptation. The EU executors on the smartphone and sensors jointly process the selected EUs, and notify the results of the applications. They execute concurrent EUs in a shared manner; the same tasks involved in multiple EUs are executed once and the results are shared.

3.2 Preparing Multiple Energy Use Options

Fig. 4 To realize the relaxed sensor association approach, multiple energy use options should be prepared for each sensing application. Such options can be prepared by exploiting the diversity in context recognition methods and sensing modalities. First, they can be generated by utilizing different combinations of sensors placed on different body positions. For instance, falling can be monitored with diverse combinations of inertial sensors on different body limbs [14]. Second, the

diversity of sensing modalities can be exploited. A heart rate monitor can utilize an ECG sensor, a SpO2 sensor, or a pulsimeter. Also, running or walking activity can be inferred from a location sensor or accelerometers [15]. Finally, multiple sensing modalities can be further fused in diverse combinations; e.g., indoor localization can be done by fusing camera, sound, Wi-Fi, and illumination [14].

To reduce the burden of developers, the system prepares multiple energy use options for widely-used contexts, e.g., activity, location, and heart rate, in advance. As for such contexts, developers only need to specify the type of contexts of interest. EnergyCordy translates the context-level specification into multiple energy use options prepared beforehand and uses them for energy coordination. For the specification, we adopt context monitoring query, from previous mobile sensing platforms [1,5]. As for other application-specific contexts, developers directly specify alternative options to the system.

3.3 Energy Use Unit (EU)

EnergyCordy abstracts the energy use of an application over distributed sensor devices as an energy use unit (EU). An EU represents a set of devices and tasks required to execute an energy use option, which accordingly determines the energy use of an application over the devices when using the option. An EU serves as the basic unit for energy allocation and scheduling. Based on the single, unified abstraction, the system can easily deal with various applications running over heterogeneous devices and flexibly apply diverse coordination policies.

Specifically, an EU, EUi,j for an application appi is defined as a set of pairs (dk, TSetki,j), where dk is a identifier of a device, and TSetki,j is the set of tasks on the device dk required to execute the EUi,j. Note that we explicitly regard inter-device data transmission as separate tasks.

For energy use coordination, the energy demand of each EU should be effectively accounted. EnergyCordy accounts the energy demand in task-level, instead of EU-level. This enables to estimate the energy use of a sensor more accurately when multiple concurrent EUs use the sensor in a shared manner. For example, when two concurrent EUs run a FFT task on the same sensor, the task-level accounting can effectively remove the overlap during the demand calculation. To reflect such potential concurrency in sensor use, the accounting is done at runtime.

For energy use accounting, EnergyCordy maintains a task-energy map, TEMapk, for each device, dk. Each map contains the average energy consumption rate for every task runnable on the device. We currently utilize an offline profiling method to build a task-energy map. The demand estimation through offline profiling works well in our environment where each task usually performs periodic operations at a fixed-time interval. Accordingly, the energy consumption for such tasks is unlikely to fluctuate over time.

Based on the task-energy map, EnergyCordy estimates the energy demands on each sensor to execute a set of EUs; the estimation is done by summing the energy demand of each task involved in the EUs as well as the base energy demand of the sensor. Given a set of EUs, P, the energy demand ed on a sensor dk is estimated as follows.

We assume that each task's energy demand and their total demand have a linear relationship. The method works well with the the sensor prototypes and tasks we used. To achieve better accuracy, it is also possible to pre-profile the energy demand of all comibnations of task sets.

$$ed = base_k + \sum(TEMap_k(task_{k,m})) \text{ s.t. } task_{k,m} \in \bigcup_{EU_{i,j} \in P} \mathrm{TSet}_k^{i,j}$$
where $base_k$ is the base energy demand of d_k

3.4 Energy Monitoring

To monitor the energy availability of sensor devices, the current prototype implementation uses the voltage-based method [18]. This method estimates the remaining energy based on voltage values. It is easy to implement since many widely used sensor devices are equipped with a voltage sensor giving real-time voltage readings. We build maps that translate voltage values to remaining energy, with which sensor voltage readings are converted to estimated remaining energy at runtime. While this method works reasonably well in our environment, the estimation accuracy can be further improved with hardware support. Elaborating on the remaining energy estimation method and improving the accuracy is beyond the scope of this paper, which is our future work.

IV. Energy Use Coordination

4.1 Coordination Policy

The energy scheduler determines the best schedule to apply the alternative EUs for each application. The key challenge is that complex inter-dependency on the use of diverse EUs for different applications. Multiple EUs might have overlap on the sensor use among each other; it is also true even on the EUs for a single application. Accordingly, the execution of an EU for an application not only affects the lifetime of the applications sharing the sensors, but its effect is propagated to all applications on the network. Also, due to the irreversible characteristics of energy resource, a naïve EU selection at a moment would continuously affect all other applications sharing the same sensors, leaving no way to redeem the energy loss by the mistrial. To address such issues, the energy scheduler carefully analyzes the spatio-temporal inter-dependency in sensor energy use among alternative EUs, and generates a schedule over the whole time. As a result, the schedule is built in the form of an energy use schedule (EUSched), which represents timelines associated with sequences of EUs to apply for all applications.

- EUSched = {EUSeqi | EUSeqi is a sequence of (EUi,m, ti,m) pairs over a timeline representing the execution sequence of EUi,m and their duration, ti,m for an application appi}

Through proper EUS generation, EnergyCordy coordinates the energy use of multiple competing applications, and prevents early shutdown of applications and sensors.

EnergyCordy can flexibly incorporate diverse coordination policies and algorithms to generate a EUSched based on the EU abstraction. In the rest of this section, we introduce three useful policies and related EUSched generation algorithms to achieve the goals defined for the policies.

4.2 Maximizing the Minimum Application Lifetime

Mobile sensing applications should run continuously to monitor a user's contexts and provide situation-aware services. Accordingly, a user will want to maximize the applications' lifetime if the next recharging time is unknown. For a policy to meet such a need of user, the primary goal of EnergyCordy can be set to maximize the minimum lifetime of any registered applications. In this case, we assume that all applications have the same priority; we address the situation that there are different priorities later in this section.

To satisfy the goal, we devise a linear programming (LP)-based method. Given a set of applications, AppSet, and a set of sensors, SSet, the method determines the optimal EUSched that best meets the goal under the energy constraints of the sensors. To find the optimal schedule, it is necessary to determine the order and duration to apply EUs for all applications. However, dealing with both duration and order as variables complicates the constraint checking for linear programming due to tasks sharing among concurrent EUs and the following energy accounting. Thus, we introduce the concept of co-execution plan Px that represents a set of EUs concurrently executed for all applications at a moment. Px not only simplifies the constraint evaluation but also changes the non-linear balancing objective to linear one, facilitating the problem formulation as below. Px can be specified as follows.

Px ⊂ { EUi,j | there must exist an EUi,j for all 1≤i≤ N,
and a ≠c for any two EUa,b and EUc,d }, for $1 \le x \le MN$, where M is the maximum number of EUs and N is the total number of registered applications.

The energy demand to execute Px is defined as a vector Ex = (ex,1, …, ex,k, …, ex,L) where ex,k represents the energy consumption rate of a sensor, sk, to execute Px. Now, the balancing problem can be formulated as follows:

$$Maximize \sum t_x \quad (1)$$

$$Subject\ to\ [E_1, \dots, E_x, \dots E_{M^N}] \begin{vmatrix} \vdots \\ \end{vmatrix} \le \begin{vmatrix} \vdots \\ \end{vmatrix}, \text{and} \quad (2)$$

$$t_x > 0,\ for\ 1 \le x \le M^N$$

, where tx is the time duration to apply Px.

As noted, the non-linear optimization goal (to maximize the minimum lifetime of applications) is transformed into the linear goal (to maximize the lifetime to run a set of Px). Since a Px contains an EU for every application, maximizing the total durations to apply Px, i.e., , guarantees to maximize the minimum lifetime. This allows applying conventional solutions for linear programming. The LP-based method can be practically applied only with a small number of applications and their EUs, e.g., about under 10, since the time complexity is exponential. Note that the problem itself is NP-hard and can be reduced from another NP-hard problem, 3-dimensional

matching problem.

To deal with a case that a user uses a large number of sensors and applications, we also develop a heuristic solution. The key idea is to arrange EUs for each application with an altruistic principle; it forces an application having diverse EU options with relatively rich energy availability to yield resources with high competition to other applications having few EU options with scarce energy resources. Each application is allowed to use resources with high competition only after it exhausts resources with low competition. This makes applications with fewer options to preferentially occupy the energy resources with high competition. Thus, the lifetime gap among the queries can be significantly narrowed.

V. Experiments and Evaluation

5.1 Prototype Development

We have implemented the smartphone-side architecture of EnergyCordy on Android SDK 2.3 and deployed it on Google Nexus One. The total LoC are about 9,500. We connect one base sensor device to the smartphone via Bluetooth to support ZigBee communication between the smartphone and sensor devices. To support various sensing applications and their energy use options, EnergyCordy incorporates 42 types of tasks commonly used for feature extraction and context recognition.

The sensor-side architecture is implemented in NesC on top of TinyOS 1.1.11 and is deployed on four types of personal sensor devices presented in Section 2.1. The LoC are about 2,500. The implementation is based on event-driven concurrency model of TinyOS. To support diverse mobile sensing applications as a general-purpose sensing platform, we implemented the functionality that allows dynamic allocation and deallocation of tasks. Each sensor task is implemented as a TinyOS event handler, including sensing, feature extraction, and status reporting. To dynamically execute the assigned tasks, we implemented a job scheduler that executes the assigned tasks in given order and frequency. The current implementation includes a set of frequency-domain feature extraction tasks, e.g., FFT, and time-domain statistical feature extraction tasks, e.g., norm and RMS. Note that a sensing module is powered on only when the corresponding sensing task is allocated for energy saving. To save highly limited sensor memory space, we implemented a shared sensing data buffer for multiple feature extraction tasks using the same sensing source. The prototype consumes about 26.2 Kbytes of program memory and 1.1 Kbytes of RAM.

(a) Common tasks

Task	Task description	Avg. energy cons. (mW)
accel	accel. sensing (3-axis, 30Hz)	2.4
gyro	gyro. sensing (3-axis, 30Hz)	20.8
norm	norm feature computation	3.3
FFT	FFT feature computation	3.0

(b) Sensor-specific tasks

Sensor ID	Task	Task description	Avg. energy cons. (mW)
d_1	*ECG*	ECG sensing (50Hz)	33.9
d_2	*weather*	weather sensing	55.1
d_3	*SpO*$_2$	SpO$_2$ sensing	164.8

(c) Basic energy consumption

Sensor ID	d_0	d_1	d_2	d_3
Basic energy consumption (mW)	37.8	18.0	18.0	37.8

(d) Transmission task

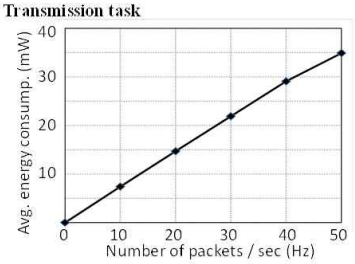

Fig. 5. Energy profiling.

5.2 Energy Use Estimation

Energy demand profiling: We first build the task-energy map for prototype sensors through offline profiling. To facilitate the profiling for diverse tasks, EnergyCordy provides an energy profiling tool. It semi-automatically measures energy consumption per unit time using an Agilent 34410A digital multimeter. A part of the results are summarized in Fig. 5. Fig. 5 (a) and (b) show the average energy consumption of common tasks on all sensors and sensor-specific tasks, respectively. We noticed slight differences in energy demand even between the same types of sensor devices. To compensate the error, we adjusted the energy demand profile for each device. Fig. 5 (c) shows the base energy consumption of each device, i.e., the energy consumed when it performs only primitive EnergyCordy operations; transmission of heartbeat and voltage messages for notifying the smartphone of its presence and energy status. Fig. 5 (d) shows the average energy consumption of transmission

tasks as the number of transmitted packets increases. We observe that the energy consumption almost linearly increases with the number of packets.

Energy availability monitoring: Fig. 6 shows voltage to remaining energy graphs we obtained from experiments. The experiments were conducted based on two different sensor devices with a rechargeable 250mAh Li-polymer battery. To evaluate the applicability of our method, we conduct experiments with a range of setting, i.e., different sensor devices running different tasks; (1) (d1, base), (2) (d1, ECG), (3) (d2, base). The graphs show the similar pattern overall although there is a little variation between them. Interestingly, voltage values do not decrease linearly. They slowly decrease when the amount of remaining energy is relatively large. When the remaining energy is smaller than 300J, the values sharply decrease.

EU execution time estimation: Table 2 shows the estimated lifetime and measured one for five different cases. We intentionally used the batteries with different initial energy to test the accuracy of our estimation in diverse potential system conditions. We measured the real lifetime of a device by executing given task set on the device until it exhausts its energy. Compared with the measured one, the estimated execution time is fairly accurate with an error of smaller than 8.2% on average. This shows that our estimation method reasonably works well. We observed that the estimation error tends to slightly increase when the amount of remaining energy is relatively large. This is because the error of remaining energy estimation is likely to increase with relatively large voltage value as discussed before.

Table 2. EU execution time estimation

Case	Device	Task set	Estimated initial energy(J)	Estimated lifetime(min)	Measured lifetime(min)	Error (%)
1	d_0	{accel,gyro, norm}	104	78.2	78.8	-0.8
2			318	238.8	202.2	18.1
3	d_1	ECG	864	432.5	409.4	5.6
4	d_2	{accel,gyro, norm}	307	230.7	215.4	7.1
5	d_3	{accel,gyro, norm,FFT}	241	130.8	119.8	9.2

We evaluate the estimation accuracy of EU execution time based on estimated energy demand and remaining energy. The execution time of an EU is determined by the minimum lifetime of the associated sensor devices. The lifetime of a device is estimated as its expected remaining energy divided by the energy demand of the EU on the device.

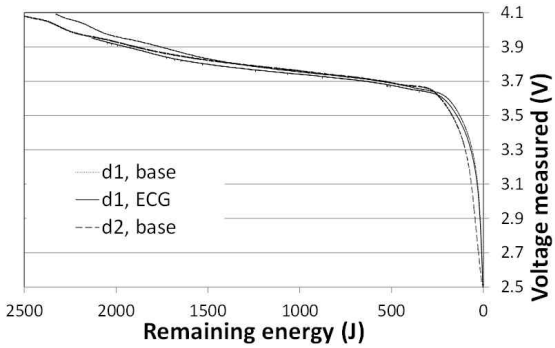

Fig. 6. Voltage-to-remaining energy map

5.3 Evaluation of the EnergyCordy Coordination

5.3.1 Experimental Setup

Energy use options: based on the prototype system and devices, we have developed the energy use options of the four example applications. Table 3 shows the EUs and their associated devices, task sets, and utility level; we represent the utility of an EU in three levels according to the recognition accuracy: high, medium, and low.

Physical activity monitor: we developed three alternative EUs using different combinations of accelerometer sensors for activity monitoring.

Fall detector: many methods have been proposed by using on-body accelerometers and gyroscopes to detect the occurrence of a fall. We implemented multiple EUs based on several of these methods using different sensor placement, sensor set, and accuracy.

Table 3. Alternative EUs of example applications

Application	EU			QoS Level
	EU ID	Device	Task set	
Physical activity Monitor	$EU_{0,0}$	d_0	{*accel, trans* (18Hz)}	High
		d_1	{*accel, trans* (18Hz)}	
	$EU_{0,1}$	d_1	{*accel, trans* (18Hz)}	Medium
		d_3	{*accel, trans* (18Hz)}	
	$EU_{0,2}$	d_0	{*accel, trans* (18Hz)}	Low
		d_2	{*accel, trans* (18Hz)}	
Fall detector	$EU_{1,0}$	d_1	{*accel, gyro, norm* (accel) *norm* (gyro), *trans* (12Hz)}	High
		d_3	{*accel, gyro, norm* (accel) *norm* (gyro), *trans* (12Hz)}	
	$EU_{1,1}$	d_2	{*accel, gyro, norm* (accel) *norm* (gyro), *trans* (12Hz)}	Medium
	$EU_{1,2}$	d_0	{*accel, gyro, norm* (accel) *norm* (gyro), *trans* (12Hz)}	Low
Heartbeat monitor	$EU_{2,0}$	d_1	{*ECG, trans* (10Hz)}	High
	$EU_{2,1}$	d_3	{SpO_2, *trans* (1Hz)}	Medium
Environmental monitor	$EU_{3,0}$	d_2	{*weather, trans* (1Hz)}	High

Heartbeat monitor: there are two major methods of heartbeat monitoring: one using ECG sensor on a chest and another using SpO2 sensor on a finger. The former usually provides better accuracy, but more expensive and less com-fortable than the latter. We developed two EUs using ECG sensor on d1 and SpO2 sensor on d3.

Environment monitor: the initial implementation of the application monitors current weather status such as temperature, pressure, and illumination. This application has only a single EU.

Alternative systems: for comparison, we implemented three alternative energy coordination systems as following:

NonAdaptive: it is a basic mobile sensing system that utilizes a single, fixed processing logic for each application that provides the best recognition accuracy or the most energy efficiency.

UseRate: it adopts the concept of the budget distribution from [19]. It distributes the energy budget of each sensor to applications, proportionally to the number of the applications of which EUs use the sensor. For example, the activity monitor acquires one half of the available energy of d1(on thigh), since only two applications use the sensor.

Separate: it represents an application-driven energy coordination system. Each application directly arranges its EUs to achieve the coordination goal, independently of other applications.

EnergyCordy setting: for all experiments, we set the initial energy of sensor devices to 3,330J; this value represents the fully-charged energy of 250mAh battery. However, it is very difficult to precisely adjust the remaining energy of the real battery. To overcome this, we obtain the remaining energy of sensor devices by subtracting the consuming energy from the initial value, instead of the estimation from the battery voltage. By default, we set the accuracy requirement of the applications to the medium level; EnergyCordy does not exploit EUs that have the low level.

5.3.2 Maximizing the Minimum Application Lifetime

We evaluate the performance of EnergyCordy with the policy that maximize the minimum of application lifetimes. Fig. 7 presents the lifetime of the applications on different systems. On this policy, the alternative systems arrange EUs to maximize the lifetime of each application, i.e., an energy efficient EU in prior to a less efficient EU. The result shows that EnergyCordy achieves more balanced lifetime than other systems. On EnergyCordy, the minimum lifetime is 554 minutes of heartbeat monitor, whereas NonAdaptive, UseRate and Separate provide 302, 197, and 404 minutes of the minimum lifetime, respectively. NonAdaptive shows most skewed lifetimes because each application has only a single EU and thus the achieved lifetime is proportional to the energy demand of the EU. Interestingly, UseRate shows much shorter lifetimes than other systems. This is because it cannot fully exploit the available energy of the sensors; it executes EUs only within the allocated budget to the applications even if the devices have remaining energy. We also evaluate the coordination effect of EnergyCordy with different utility requirements; High, Medium, and Low. Fig. 8 depicts the results. Regardless of the utility level, the lifetimes of the applications are well balanced on EnergyCordy. The minimum lifetime for High, Medium, and Low is 473, 554, and 592 minutes, respectively. As the requirement lowers, the achieved lifetime of applications increases. This is because lower requirement allows EnergyCordy to utilize more EUs and thus provides more opportunities in coordinating the applications' energy use.

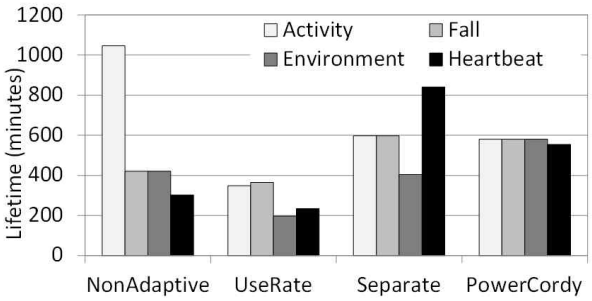

Fig. 7. Lifetime of applications

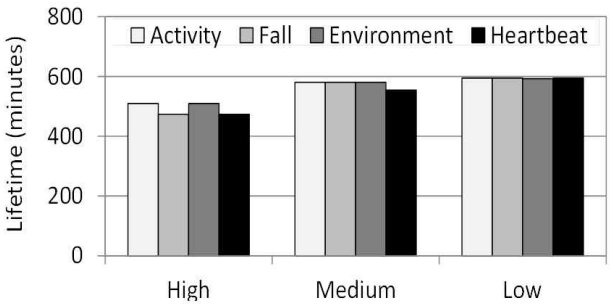

Fig. 8. Lifetime with different utility requirement

VI. Conclusions

In this paper, we propose EnergyCordy, a novel inter-device energy use coordination system; with a system-wide holistic view, it coordinates energy use over multiple sensor devices by concurrent sensing applications. As a key approach, we propose a relaxed sensor association. It decouples the energy use of an application from specific sensor devices, allowing flexible contention resolution at runtime. More specifically, it prepares multiple energy use options (EU) that use different sensor combinations, and schedules the order and duration to apply EUs for coordinated energy use. We demonstrated the effectiveness of EnergyCordy by developing multiple example applications over diverse custom-designed senor devices.

The result shows that EnergyCordy achieves much more balanced lifetime than other systems such as NonAdaptive which shows most skewed lifetimes because each application has only a single EU and thus the achieved lifetime is proportional to the energy demand of the EU. Regardless of the utility level, the lifetimes of the applications are also well balanced on EnergyCordy. We conclude that EnergyCordy effectively coordinates energy use of concurrent applications over multiple sensor devices and prevent undesired early shutdown of an application.

REFERENCES

[1] Nicholas D. Lane, Emiliano Miluzzo, Hong Lu, Daniel Peebles, Tanzeem Choudhury, and Andrew T. Campbell, “A Survey of Mobile Phone Sensing,” IEEE Communication vol. 9, no. 5, pp. 686-702, May 2010.

[2] Ozgur Yurur, Chi Harold Liu, Zhengguo Sheng, Victor C. M. Leung, Wilfrido Moreno, and Kin K. Leung, "Context-Awareness for Mobile Sensing: A Survey and Future Directions", IEEE Communications Surveys & Tutorials, vol. 18, issue 1, pp. 68-93, 2016.

[3] Takamasa Higuchi, Hirozumi Yamaguchi, and Teruo Higashino, "Mobile Devices as an Infrastructure: A Survey of Opportunistic Sensing Technology", Journal of Information Processing, vol. 23, no. 2, pp. 94-104, 2015.

[4] Muhammad Shoaib, Stephan Bosch, Ozlem Durmaz Lncel, Hans Scholten, and Paul J. M. Havinga, "A Survey of Online Activity Recognition Using Mobile Phones", Sensors, vol. 15, no. 1, pp. 2059-2085, 2015.

[5] V. Agarwal, N. Banerjee, D. Chakraborty, and S. Mittal, “USense - A Smartphone Middleware for Community Sensing,” IEEE International Conference on Mobile Data Management, volume 1, pp. 56-65. IEEE, June 2013.

[6] K. Lorincz, B.-r.Chen, G.W. Challen, A.R. Chowdhury, S. Patel, P. Bonato, and M. Welsh, “Mercury: a wearable sensor network platform for high-fidelity motion analysis,” Proc. SenSys, 2009.

[7] U. Maurer, A. Smailagic, D.P. Siewiorek, and M. Deisher, “Activity recognition and monitoring using multiple sensors on different body positions,” Proc. BSN, 2006.

[8] Y. Lee, Y. Ju, C. Min, S. Kang, I. Hwang, and J. Song, “CoMon: Cooperative Ambience Monitoring Platform with Continuity and Benefit Awareness,” Proc. MobiSys, 2012.

[9] K. Lin, A. Kansal, D. Lymberopoulos, and F. Zhao, “Energy-accuracy trade-off for continuous mobile device location,” Proc. MobiSys, 2010.

[10] Angel, the open sensor for health and fitness. http://angelsensor.com/

[11] Sensordrone. http://sensorcon.com/sensordrone/

[12] Y. Fei, L. Zhong, and N.K. Jha, “An energy-aware framework for dynamic software management in mobile computing systems,” ACM Transactions on Embedded Computing Systems, vol. 7, issue 3, article no. 27, April 2008.

[13] A. Lachenmann, P.J. Marron, D. Minder, and K. Rothermel, “Meeting lifetime goals with energy levels,” Proc. SenSys, 2007.

[14] M. Azizyan, I. Constandache, and R.R. Choudhury, “SurroundSense: mobile phone localization via ambience fingerprinting,” Proc. MobiCom, 2009.

[15] L. Bao and S.S. Intille, “Activity recognition from user-annotated acceleration data,” Proc. Pervasive, 2004.

[16] M. Budde, R.E. Masri, T. Riedel, and M. Beigl, “Enabling low-cost particulate matter measurement for participatory sensing scenarios,” Proc. Int’l Conf. on Mobile and Ubiquitous Multimedia (MUM), 2013.

[17] Q. Li, J.A. Stankovic, M.A. Hanson, A.T. Barth, J. Lach, and G. Zhou, “Accurate, fast fall detection using gyroscopes and accelerometer-derived posture information,” Proc. BSN, 2009.

[18] A. Lachenmann, P.J. Marron, D. Minder, and K. Rothermel, "Meeting lifetime goals with energy levels," Proc. SenSys, 2007.

[19] H. Zeng, C.S. Ellis, A.R. Lebeck, and A. Vahdat, "ECOSystem: managing energy as a first class operating system resource," Proc. ASPLOS-X, 2002.

Authors

Yunseok Rhee received the B.S. degree in Computer Science and Statistics from Seoul National University, Korea in 1984 and the Ph.D degree in Computer Science from KAIST, Korea in 1999 respectively.

He is currently a Professor in the Division of Computer and Electronic Systems Engineering at Hankuk University of Foreign Studies. His current research interests include context-aware systems, embedded systems, and internet services.