Future Generation Computer Systems 39 (2014) 44–54

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Energy efficient indoor tracking on smartphones

Dezhong Yao[a,b], Chen Yu[a,*], Anind K. Dey[b], Christian Koehler[b], Geyong Min[c], Laurence T. Yang[a], Hai Jin[a]

[a] Services Computing Technology and System Lab, Cluster and Grid Computing Lab, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, 430074, China

[b] Carnegie Mellon University, Pittsburgh, 15217, United States

[c] School of Informatics, University of Bradford, Bradford, BD7 1DP, UK

HIGHLIGHTS

- We profile the energy consumption of different computing and communication use cases.
- An accelerometer-based method for adaptively managing the use of the WiFi adapter is proposed.
- We propose a sensor management strategy for adaptive duty cycling and a data upload strategy for indoor situations.

ARTICLE INFO

Article history:
Received 2 March 2013
Received in revised form
2 November 2013
Accepted 13 December 2013
Available online 22 December 2013

Keywords:
Mobile computing
WiFi-location
Energy-aware systems
Motion learning

ABSTRACT

Continuously identifying a user's location context provides new opportunities to understand daily life and human behavior. Indoor location systems have been mainly based on WiFi infrastructures which consume a great deal of energy mostly due to keeping the user's WiFi device connected to the infrastructure and network communication, limiting the overall time when a user can be tracked. Particularly such tracking systems on battery-limited mobile devices must be energy-efficient to limit the impact on the experience of using a phone. Recently, there have been a lot of studies of energy-efficient positioning systems, but these have focused on outdoor positioning technologies. In this paper, we propose a novel indoor tracking framework that intelligently determines the location sampling rate and the frequency of network communication, to optimize the accuracy of the location data while being energy-efficient at the same time. This framework leverages an accelerometer, widely available on everyday smartphones, to reduce the duty cycle and the network communication frequency when a tracked user is moving slowly or not at all. Our framework can work for 14 h without charging, supporting applications that require this location information without affecting user experience.

1. Introduction

Many useful applications require long-term, high-accuracy tracking of the location of their users while indoors. These include wayfinding applications for people with disabilities [1–3], indoor monitoring of elders [4], locating people in emergency response situations, augmented reality and point-of-sale applications. However, the most accurate indoor tracking systems require infrastructure to be purchased and deployed, and for users to wear tags or beacons [5,6]. Instead, researchers are relying on infrastructure that is already commonly available: WiFi networks with multiple access points and WiFi-enabled smart phones carried by users. While there are many interesting issues to be addressed in improving the accuracy of localization with such systems, here we focus on how to improve their energy-efficiency. Many applications require real-time tracking and/or long-term history, which necessitates energy-efficient localization. But the WiFi adapter on smart phones consumes a lot of energy, limiting the usage time of mobile devices [7]. There are two types of long-term tracking applications: One type is applications that may not need a high sampling rate and data communication frequency, such as wayfinding applications. The other type is applications that need a high sampling rate, like emergency tracking systems. Current studies do not provide models or algorithms for finding the optimal sampling rate and data communication frequency for different types of applications.

The main goal of this paper is to build an energy-efficient localization framework that automatically manages sensor availability, accuracy and energy. There is an important tradeoff between localization accuracy in indoor environment and battery lifetime. For wayfinding and health monitoring applications, they need the longest battery lifetime possible and do not require the highest accuracy. Some navigation systems need to have high accuracy, but

* Corresponding author. Tel.: +86 2787541924x8023.
E-mail addresses: yuchen@hust.edu.cn, hust.yuchen@gmail.com (C. Yu).

0167-739X/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.future.2013.12.032

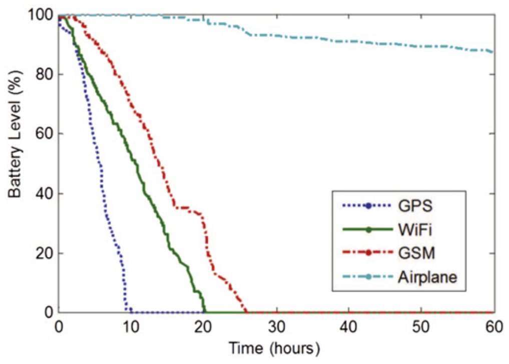


Fig. 1. Energy consumption comparison.

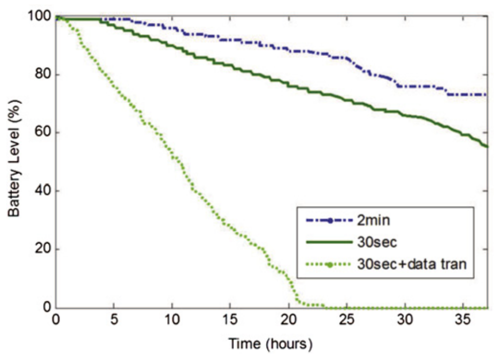


Fig. 2. Sensing interval comparison for WiFi.

are not very concerned with energy consumption as they are often not used for a long period of time. For our daily tracking applications, we need to build a model to extend the battery life as long as possible, while also dynamically meeting the accuracy goal at the same time. From a user experience perspective, this model needs to allow the system to optimize battery life by intelligently managing the location accuracy and energy trade-offs based on available sensors. To realize the above goal, we developed an approach based on two observations. First, location applications do not always need the highest available accuracy, such as when people are standing in one spot for a long time or going down or up stairs. Second, a phone has multiple modalities to sense indoor location aside: WiFi triangulation [8,9], cell-tower triangulation [10], Bluetooth vicinity, audio-visual sensing [11], accelerometer sensing. Those modalities can be selected to efficiently meet the location needs at lower energy costs. In another part, we can use energy prediction method [12,13] to calculate how much energy will be used in the next few minutes based on current motions of a user.

To explore the framework, we first use a modular approach to build a WiFi-based indoor location system based on the Android platform and an existing Wifi infrastructure. Then, a detailed measurement study is conducted to quantify the energy consumed by the different modules. We compare the energy costs of localization when running a localization algorithm locally on the mobile device and on the remote server. We find that energy consumption is intimately related to the data transmission time and scan frequency of the WiFi adapter. Executing the location algorithm locally on the smart phone minimizes the use of the WiFi adapter, and thus produces the most energy-efficient results.

The work presented in this paper makes the following contributions: (1) we profile the energy consumption of different computing and communication use cases to design an energy-efficient indoor tracking framework; (2) we propose an accelerometer-based method for recognizing a user's movement status for adaptively managing the use of the WiFi adapter; (3) we propose a sensor management strategy for adaptive duty cycling and a data upload strategy for indoor situations; (4) we demonstrate that the framework, with the proposed strategies, can be implemented on the Android platform with a battery life of 14 h in the presence of regular phone usage, an improvement of 4 h over a static duty cycle approach. The evaluation results show our framework can handle different activity modes, *e.g.*, standing, walking, going upstairs/downstairs. Furthermore, it can support whole day indoor tracking without a significant impact on a smart phone's battery life.

The rest of the paper is organized as follows: The challenges of indoor energy-efficient tracking are explained in Section 2. The related work is also summarized in Section 2. In Section 3, we present the detailed design of our framework. Section 4 describes the framework architecture and our implementation method including the implementation of the motion monitor. The framework's performance is evaluated in Section 5. Finally, Section 6 concludes with a discussion of our experiments and framework.

2. Motivation

In this section, we motivate the work by highlighting the results from a set of experimental evaluations. We describe the factors impacting energy efficiency in indoor location sensing through an initial experiment with Android smart phones and summarize the limitations of existing outdoor energy-efficient location-sensing approaches.

2.1. Energy hungry device: WiFi adapter

We first estimate the impact of using the power intensive WiFi adapter on smart phones. By considering a scenario where a user is working in a building with a location tracking application (LTA) running. The application determines the user's current location in a timely manner using the Android location API. The battery usage level is measured using the Android power API. For comparison, we run the same LTA on the same phone in 4 different settings: in airplane mode, WiFi only, cellular network only, and GPS only. The location refresh rate is set to 30 s and each experimental setting is stopped when the phone turns itself off due to the battery dying.

Fig. 1 depicts the battery level of the phone for each experimental setting. As shown in Fig. 1, GPS consumes more power than the other localization technologies. Using WiFi and GSM, the location tracking application can work for more than 20 h without other activities. WiFi and GSM signals are commonly available in indoor locations. But these methods also use too much energy when the location application is continuously running, limiting the use of other applications.

2.2. Data transform compression

WiFi has two main energy consuming components: (1) scanning and associating to an access point (AP) and (2) transferring data [14]. We conduct experiments to understand the relative impact of these aspects. Consider two conditions for a location tracking application (LTA) that uses WiFi-based location. In the first condition, it only samples the WiFi signal strength with a scan interval of 30 s, but does not transmit any data. The second condition is the same as the first, but it also transmits data to a server, by having the WiFi adapter connect to an AP. We can see the impact on battery lifetime of these two conditions in Fig. 2. The energy consumed by data transmission is nearly five times than that of the scan and association. Other work has shown that transmitting larger amounts of data does not consume significantly more power, and can reduce the frequency of uploading data to a remote server [14,15], reducing overall energy consumption. However, this approach has limited applicability as real-time needs for localization information means data needs to be available in a timely manner.

2.3. Sensing intervals

For many mobile platforms, we cannot set the length of each scanning pass. However, it is possible to control the time interval between each scan. Intuitively, longer time intervals will help save energy [7]. To study the impact of adapting sensing intervals, we consider our LTA with a scan frequency of 2 min. As shown in Fig. 2, by simply enlarging the scan interval from 30 s to 2 min, energy consumption is reduced by 30%.

2.4. Problem characterization

We now summarize these issues to better characterize the problem of energy-efficient indoor tracking and our proposed long-term sensing solution.

1. *Accuracy*: Location-tracking applications will need guarantees that the requested level of accuracy will be met. An energy-efficient system will need to make reasonable tradeoffs between energy usage and tracking accuracy.
2. *Use local computing resources*: Smart phones have powerful computing and storage capacities. A location sensing algorithm and database can be executed entirely on such platforms, reducing the energy spent on transporting data to a remote server. Only when the server requires location information does the data need to be transported.
3. *Dynamic data update rate*: In many cases, data updates to the server occur at a fixed interval. However, sometimes the data is not meaningful (i.e., not different from the previous value), so there is no need to transmit it. Dynamically adapting the update rate based on accuracy requirements and changes in the data can help to reduce energy consumption.
4. *Use context model to optimize location-sensing*: By leveraging specific power-efficient sensors from smart phones, such as the accelerometer and orientation sensors, additional context can be obtained. In particular, human activity can be recognized and used to save energy. When a user is stationary or moving within his/her office, we can reduce the sampling rate in order to save energy.

2.5. Related work

2.5.1. Energy-efficient location tracking

Many systems have been proposed to improve energy efficiency in position systems. They mainly focused on sensor management strategies for determining when to deactivate energy hungry sensors or optimizing sensor duty cycles by selecting sensors with low power consumption whenever possible. An example is EnTracked [15], which attempts to control the GPS duty cycle for determining location. It schedules position updates by using an accelerometer-based mobility estimation to save energy and was evaluated only in simulation. Farrell et al. [16] present an algorithm that determines when to perform GPS updates, based on a positioning uncertainty model. However, their model too has only been evaluated in simulation. Cloud computing [13] is a new way to address this problem, but it cannot optimize the energy usage for each sensor.

Several other researchers have explored the methods that tradeoff accuracy for energy savings. RAPS is a position tracking system used in urban areas [17]. The system motivation is that GPS positioning tends to be inaccurate and often unavailable in these areas. RAPS trades off location accuracy for reduced energy use. It uses a combination of location history, user movement, and GSM information to selectively activate GPS only when necessary to reduce position uncertainty. It also proposes sharing position readings among nearby devices using Bluetooth in order to further reduce GPS activation. However, RAPS is mainly designed for pedestrian use, and a significant portion of the energy savings comes from avoiding GPS activation when it is likely to be unavailable. EnLoc [18] uses dynamic programming to find the optimal localization accuracy for a given energy budget and decides which one of GPS, WiFi or GSM localization methods to use. MicroBlog [19] exploits the accuracy energy tradeoff of GPS, WiFi and GSM based localization for energy-aware localization. Specifically, depending on the accuracy requirement of the application, it uses a lower-energy method over a high-energy method. ALoc [20] proposes to dynamically trade-off location accuracy and energy use, based on probabilistic models of user location and sensing errors. These models are used to choose the best among different localization methods and tune energy expenditure to meet location accuracy requirements specified by an application.

Recently, tracking systems which use trajectory history have been proposed. The StarTrack middleware [21] continuously tracks users for smartphone applications. It uses a track abstraction and employs similarity matching algorithms to find mobile users who share the same routes. Mikkel et al. [22] monitors whether the tracked objects have traveled a certain distance by using sensors. They do not analysis user's activities. Escort [23] is a positioning system that obtains cues from social encounters and leverages an audio beacon infrastructure. However, these systems do not address how to collect trajectories in an energy-efficient manner. The popular commercial indoor location system, Skyhook, uses WiFi for positioning and uses energy-efficient techniques to obtain a battery life on mobile devices of 16 h. However, its accuracy guarantee is quite a bit lower than ours, at 30 m [18]. If an application needs longer battery life and can accept the reduced accuracy, Skyhook is a reasonable alternative to our proposed approach.

In comparison to earlier work, the sensor management strategies and location updating protocols present in this paper significantly reduce battery consumption. We take advantage of monitoring user's activities to reduce the need to sample power hungry sensors and to find the optimal location update interval, all while maintaining high accuracy in tracking.

2.5.2. Motion detection

Several pieces of prior work have used accelerometer-based motion detection in location systems in different ways. Kim et al. [24] use motion detection to save energy: after a user has arrived at a place and enough scans have been collected, the WiFi radios are turned off. Once the motion variance exceeds its threshold, WiFi scanning is resumed. Their goal is to find contextual information about human's locations as places and paths. Shafer and Chang [25] detect movement and, if walking is detected, perform a ''full localization'', which presumably entails taking many scans over a short period after truncating the scan queue. If a user walks for a period of time longer than the movement detection period, this can result in significantly more battery drain than our approaches because a long series of scans will be repeatedly discarded. Ledlie et al. [26] aims at a more flexible and scalable point in the design space of localization systems. All of this related work does not consider the problem of data collection throughout an entire day. Current studies are aiming to use less energy, but not to optimize tracking accuracy according to the current battery life in order to maximize tracking time. As such, we suggest that most of those works are not suitable for the location tracking applications that users will want to use in indoor environments.

3. Framework design

To support an energy-efficient localization mechanism for location tracking applications, we provide an energy consumption model and present two guiding principles.

3.1. WiFi location

A fingerprint-based localization approach is adopted in this work. We collected WiFi signal strength data across an entire building on our campus multiple times, using a 1 m × 1 m grid. We then built a signal strength map (WiFi fingerprint) containing the means and standard deviations at each grid point. To locate a user in real-time, our WiFi Localization algorithm uses Monte Carlo Localization (MCL) [27] with Bayesian filtering to maintain a set of hypotheses of the user's location. The MCL method with Bayesian filtering can be separated into two models: the perceptual model $P(S|l)$ and the motion model $P(l_t|L_{t-1}, u_{t-1})$ of the user. The perceptual model is used to calculate the probability of obtaining a particular signal strength measurement S at a location l.

By fusing the results of the perceptual model, accelerometer data and direction of the user movement, we can build a motion model to calculate a location belief $B(l)$ for location l. With an accelerometer's help, we can detect how many steps the user walked from location l_{t-1} to l_t. From the previous two recent samples, we can calculate the direction θ the user is moving in. Then the walking distance $d(t)$ can be obtained. The details for our method are presented in Section 3.4.

The term of motion model is obtained using the following equation:

$$P(l_t|l_{t-1}, d(t)) = \begin{cases} 1, & \text{direction change} \\ \dfrac{1}{\sigma_{acc}\sqrt{2\pi}} \exp\left(-\dfrac{(x'_t - x_t)^2 + (y'_t - y_t)^2}{s\sigma^2_{acc}}\right), & \text{otherwise.} \end{cases} \quad (1)$$

A change in the direction of motion or a change in motion can be detected by the accelerometer using the method we will describe in Section 3.4.

3.2. Energy consumption measurement

To efficiently track a user, we require an energy consumption model that can describe the relationship between power consumption and an accuracy requirement (i.e., how frequently a remote server/application requires location updates). The proposed energy consumption model consists of two main parts: a power model that describes the power usage for WiFi localization, and a delay update model that describes the data transmission to a remote server. Our model has the following features:

1. *Tracking accuracy* (E_{track}): required accuracy distance of an application, indicating the maximum distance a user can travel between two sampling instances.
2. *Walking speed* ($v(t)$): user walking speed m/s at time t.
3. *WiFi scan time interval* (Δt_s): time delay between each scan.
4. *Consumption of WiFi scan* (P_{scan}): energy spent in each scan cycle.
5. *Delay interval* (t_{update}): time interval between each data upload.
6. *Consumption of data update* (P_{update}): energy spent in each data upload to the remote server.

We model the energy cost for indoor location tracking for the two functions as defined in Eq. (2): The total power consumption $Power(T)$ in time T is the sum of the power spent on $Power_{scan}$ and $Power_{update}$ with a fixed scan time interval Δt_s.

$$Power(T) = Power_{scan}(T) + Power_{update}(T) \quad (2)$$

$$Power_{scan}(T) = \sum_{t=1}^{T} \left(\frac{1 * v(t) * P_{scan}}{E_{track}}\right) \quad (3)$$

$$\Delta t_s = \frac{E_{track}}{v(t)} \quad (4)$$

$$Power_{update}(T) = \frac{T}{t_{update}} * P_{update}. \quad (5)$$

As WiFi scanning and association consumes nearly five times the energy of transferring data [14,28], we need a dynamic scanning mechanism that provides a wakeup and sleep strategy for the WiFi adapter to reduce energy consumption.

3.3. Comparing communication strategies

There are lots of indoor location tracking and navigation applications which need continuous location information [29]. Some applications running locally on a mobile device need to know the device's location, and some of them need to send the trajectory data to external devices or remote services to finish monitoring tasks. Different applications have different system architectures, and that will consume different amounts of energy. As shown in Fig. 3, we use different combinations of client and server modes to implement a location tracking system. We experimented with these combinations to understand their respective energy consumptions, and show the results in Fig. 4.

1. All run locally: The WiFi fingerprint map is stored on the mobile phone and the localization algorithm also runs on it (Fig. 3(a)). In this setting, the WiFi adapter does not need to transmit data between the mobile phone and a server.
2. Logging user's trajectory: In this setting, the mobile phone executes the localization algorithm, and only communicates the user's position to a server in a timely manner (Fig. 3(a)).
3. Logging sample and trajectory: This setting performs localization on the mobile phone, but also shares the user position and raw WiFi RSS (received signal strength) samples with the server (Fig. 3(b)).
4. Return result to mobile phone: In this setting, the mobile device passes the RSS samples to a server. The server has the fingerprint database, computes the user's position, and returns the position result to the mobile device (Fig. 3(b)).
5. All stored on server: In this setting, the mobile device sends the WiFi RSS samples to the server. The server calculates the position and logs all information, with no return communication back to the mobile phone (Fig. 3(b)).

3.4. Sensor management strategies

Our framework provides three strategies for energy-efficient localization: (1) motion detection, (2) speed-aware, and (3) data upload. The strategy with the lowest energy estimate is executed. The parameters for all strategies are selected based on localization accuracy requirements (E_{track}) of the application needing location tracking and the current activity of the user. In coordination with each other, the motion detection strategy and the speed-aware strategy choose a sleep period and determine when to use a different sampling frequency for tracking the user. Based on the user's current motion and speed history, the upload data strategy will calculate when to send location records to a server to meet the applications accuracy requirement. The flow control strategy of our framework is shown in Fig. 5. The first step is to check the

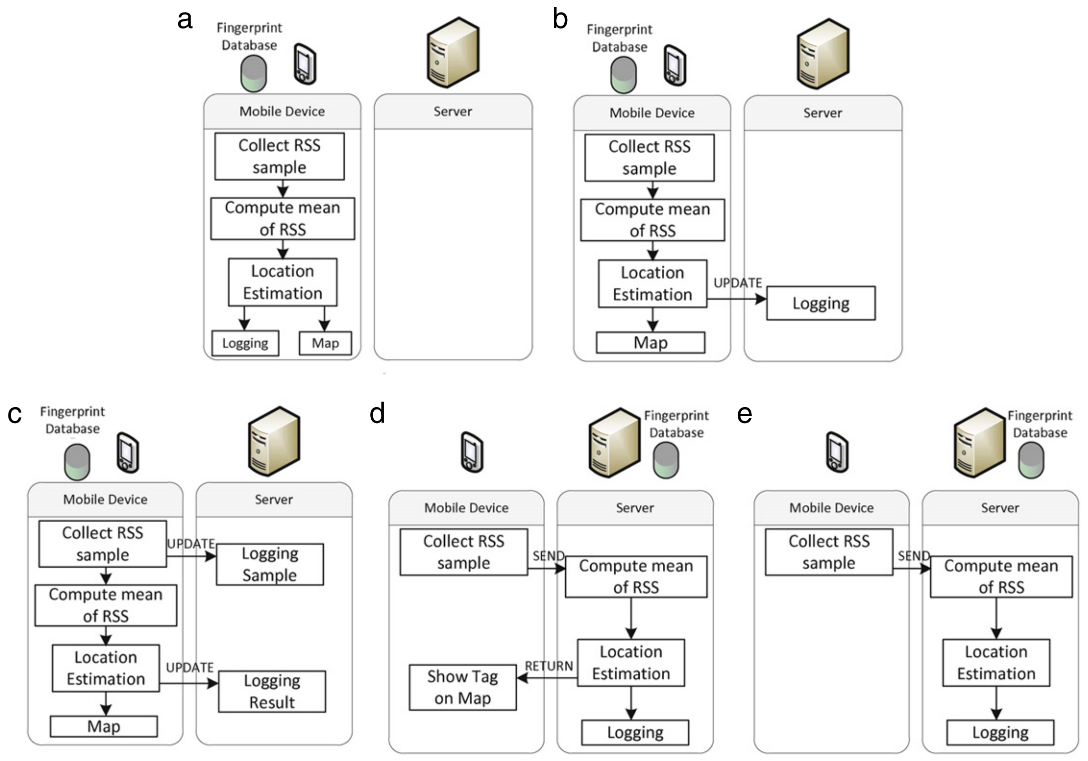


Fig. 3. Interactions between the server and mobile device.

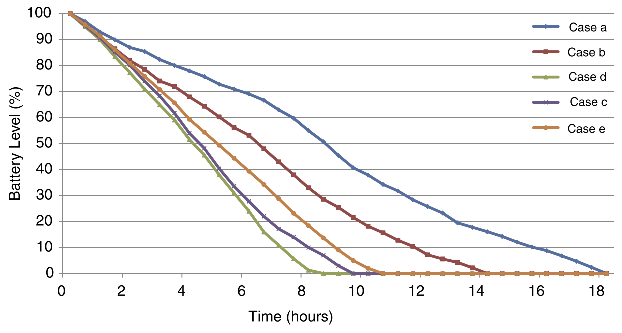


Fig. 4. Lifetime comparison for each interaction method.

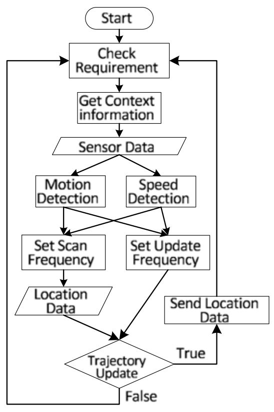


Fig. 5. Flow of control strategy.

application's accuracy requirement, followed by the collection of context for input to the energy-efficiency strategies.

3.4.1. *Motion detection strategy*

The Android platform has a default policy to turn off the WiFi when the screen turns off. But for location applications, we need to track location even when the screen is off. Design goals of our motion detector are: (1) low-power usage, (2) robust detection regardless of orientation, and (3) low tolerance to movement. We implemented a motion detection method to track a user's motion using the phone's built-in 3-axis accelerometer. Using the orientation sensor and magnetic sensor on the mobile phone, we can calculate the acceleration along the vertical and horizontal axes when the mobile phone changes orientation. $x(t), y(t)$ are the orthogonal coordinate values in the horizontal axis direction and $z(t)$ follows the vertical axis direction. Theoretically, the moving speed and distance can be obtained by integrating the acceleration signal. But for indoor pedestrians, it is challenging to infer walking distance from acceleration due to issues of offset drift and tilt variation. An alternative approach is to detect the walking pattern. When people walk, the vertical acceleration fluctuates periodically. This periodical signal can represent the steps people walked, as shown in Fig. 6. So we need to identify the fluctuation pattern and fluctuation frequency to obtain the walking distance and walking speed.

$$d(t) = StepSize \times Num\_Fluctuation(Steps) \tag{6}$$

$$Freq\_z = \max(FFT(z(t))). \tag{7}$$

In our paper, the Fast Fourier Transform (FFT) algorithm is used to analyze the acceleration value for recognizing the walking pattern and inferring the number of steps taken. We know that the vertical acceleration signal is a fixed cycle fluctuation that occurs when people are walking and the vertical acceleration signal crosses the zero line twice every step. Hence, we can count the number of zero crossing points and divide it by two, deriving the number of walked steps *Num_Fluctuation*(*Steps*). The step size is usually 0.5 m. The vibration frequency of a human is around 0.7–3 Hz [30], when walking. After we have the walking distance, we can calculate the moving speed: $v(t) = d(t)/t$.

To detect more activities from people, we use the improved normalized signal magnitude area (SMA) [31] to detect human

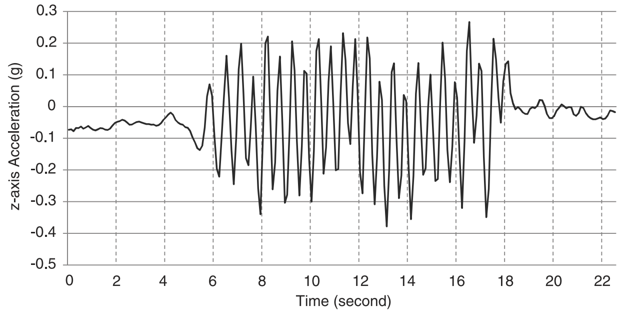


Fig. 6. Filtered acceleration in z-axis. (The offset g in z-axis is already compensated.)

Table 1
Human activities recognize.

Pre-State	SMA > 2.5 0.7 < SMA < 3.0	2.5 < SMA < 1.5 0.7 < SMA < 3.0	Other
Standing	Walking	Walking	Standing
Walking	Turning	Walking	Standing
Turning	Turning	Walking	Walking

movement speed and direction changing events. Defined in Eq. (8), SMA was used as the basis for identifying periods of activity/motion:

$$SMA = \frac{1}{t}\left(\int_0^t |x(t)|dt + \int_0^t |y(t)|dt + \int_0^t |z(t)|dt\right). \quad (8)$$

When a user starts to walk, stop or turn a corner, there is a significant acceleration in the horizontal direction, but small changes in the vertical direction. So the start of walking, stopping walking and turning events can be detected.

Calculation of the parameters in Eq. (8) is performed by summing each sampled valued progressively over a 1 s interval. An appropriate threshold value, see Table 1, was determined through a pilot study with three users. With knowledge of the previous state (Pre-State) and the current $\max(FFT(z(t)))$ and SMA value, we can classify whether the user is standing still or moving. If the user is standing still, the WiFi adapter is set to sleep. Otherwise, the WiFi adapter is turned on. Our motion detection system has an accuracy of 90%.

Knowing whether a tracked person is going upstairs or downstairs, is also an important context, which can help to reduce the WiFi scan frequency. As going upstairs or downstairs is usually a series of vertical movements and WiFi signal strength tends to be weak in stairwells, we can just use the accelerometer to track a user's location to improve the accuracy. Using an empirically determined threshold of 0.8 m/s^2 in the horizontal direction, and a direct current (DC) value about 10–20 in the FFT result of the vertical acceleration, we can conclude whether the user is going upstairs (median value of vertical acceleration is above a positive vertical threshold) or downstairs (below the negative threshold). Assuming the user moves with uniform speed, we can calculate the distance the user is moving up (or down): $d_z = \int avg\_acc_z * \Delta t$, where Δt is the time between acceleration samples that exceed the threshold. As the height of each floor is about 5 m, we can calculate how many floors the user has traversed.

3.4.2. *Speed-aware strategy*

With the user's current speed and the given accuracy requirement, the speed-aware strategy predicts the interval Δt_s between each WiFi signal scan. To determine the value of Δt_s, we use the energy consumption model described in the previous section. To minimize the energy consumption and to calculate Δt_s, the current activity information and SMA value are used. The current battery level in the smart phone is $Power_{left}$ (minimum amount of power user wants to have) and the accuracy requirement, set by the user or application is E track meters. From Eqs. (3) and (4), we derive Eq. (9). The energy used in a WiFi scan is:

$$Power_{scan}(T) = \sum_{t=1}^{T} \frac{1 * P_{scan}}{\Delta t_s}. \quad (9)$$

We assume the user is moving uniformly, so the power consumption can be calculated as.

$$Power_{scan}(T) = T * \frac{P_{scan}}{\Delta t_s}. \quad (10)$$

If we want our smart phone to continue to work for T hours, we need $Power_{scan}(T) < Power_{left}$. From Eq. (4), Δt_s has a maximum threshold due to the accuracy requirement, then:

$$\frac{E_{track}}{v(t)} > \Delta t_s > \frac{T * P_{scan}}{Power_{left}}. \quad (11)$$

So Δt_s should be less than $\frac{E_{track}}{v(t)}$, where $v(t)$ is the user's current speed and E_{track} is the current required accuracy distance. The Android platform warns the user when the battery is lower than 10% ($Power_{alert}$). In our framework, we try to avoid reaching this level:

$$\Delta t_s = \frac{T * P_{scan}}{Power_{left} - Power_{alert}}. \quad (12)$$

To understand the trade-off for energy saving, we can calculate the tracking accuracy that the current battery level will support:

$$E'_{track} = v(t) * \Delta t_s = \frac{v(t) * T * P_{scan}}{Power_{left} - Power_{alert}}. \quad (13)$$

When $E'_{track} \geq E_{track}$: it means that the current battery can only support E'_{track} accuracy requirement, so we need to use the scan interval from Eq. (12).

When $E'_{track} < E_{track}$: it means that current battery can support much more accuracy than the system requirement. However, we use Eq. (4) to get scan interval, in order to save energy.

3.4.3. *Data upload strategy*

If we store the fingerprint database on the mobile phone, our location algorithm can execute locally to get the user's current location. This approach consumed the least energy (Figs. 3 and 4, Case a). However, it is often the case that a remote server or application needs location updates from the phone (Figs. 3 and 4, Case b). Using the user's motion context information, our framework can dynamically change the upload frequency without violating the application's accuracy requirement. Given knowledge of the user's current speed pattern and accuracy threshold, the data upload strategy can determine when to upload the log of locations to the server. Our framework calculates the accumulated distance traveled by the user: d. If d is above the accuracy requirement or a moving event is detected (activity status changes), our framework will upload the trajectory dataset.

4. Framework architecture and implementation

In this section, the software architecture of our framework is presented. We explain the details of our implementation on Android smart phones.

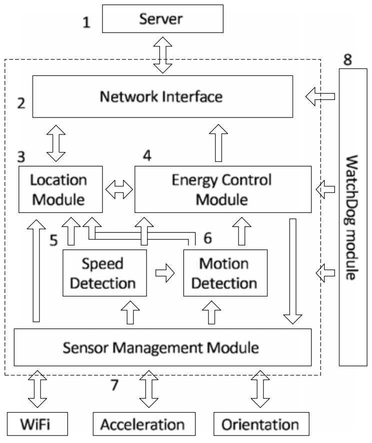


Fig. 7. Framework architecture.

4.1. Architecture and implementation

We implemented the framework (Fig. 7) in Java on an Android smart phone. The framework was implemented on Android OS versions 1.6, 2.1, 2.2 and 2.3. In most of our case studies, we used a HTC G1 smart phone. The Android phone comes with several built-in sensors, including GPS, WiFi adapter, accelerometer, and orientation sensor. Our framework controls the state of the WiFi adapter. The phone acquires the WiFi signal strength from the WiFi adapter, calculates its location locally and updates the server in a timely fashion (Fig. 3(a)). The framework has four main modules: Energy Control Module, Location module, Sensor Management module and Watchdog module. The framework also has other components for motion detection, speed detection and sensor management.

In our case study, the motion detection component uses the acceleration and orientation sensors' data to calculate the acceleration $x(t)$, $y(t)$ and $z(t)$ value in three directions. Using the classification algorithms on this processed data, the motion detection component returns user activity features such as "standing", "walking", "running", "going upstairs" and "going downstairs". The speed detection component also gathers 3-axis accelerometer sensing data to compute the user's current walking speed. Based on the real-time walking speed, we can measure how far the user has walked. With this result, we can adjust the WiFi scanning interval. With the results from the speed detection and motion detection components, the energy control module configures the WiFi on/off, scanning interval and update rate.

Our framework does not have a Graphic User Interface (GUI) to show information to user. All of the modules in Fig. 7 are services, running in the background that are woken up and kept alive by the Watchdog module. The role of each component is: (1) The remote server stores the fingerprint database and runs the location algorithm. (2) Network interface has a buffer to store location data results. (3) Location Module is in charge of gathering WiFi signal strength and movement. (4) Energy Control Module configures the framework based on the context environment. (5) Speed Detection component gives the real-time walking speed. (6) Motion Detection component detects user activity status. The Sensor Management module listens to the Android OS broadcast messages when the sensors' status changes. (7) Sensor Management Module configures sensors and read sensor data. (8) The Watchdog module wakes up the other modules when they go to sleep.

4.2. Motion monitor

After detecting user motion, we need to configure our WiFi scan frequency depending on the activity being performed. Fig. 8

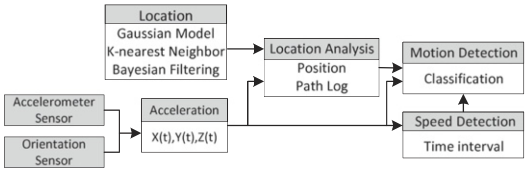


Fig. 8. Motion detection implementation.

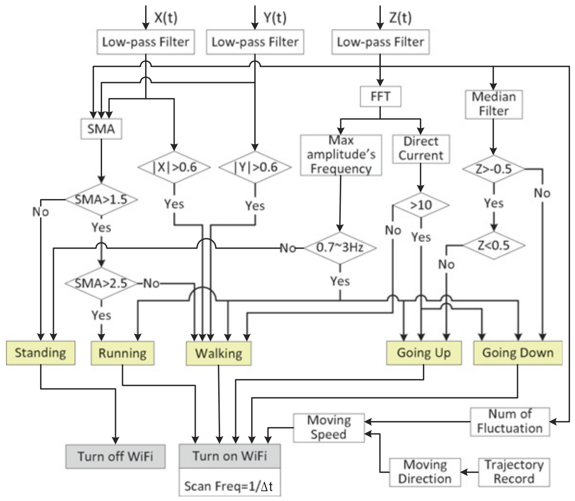


Fig. 9. Real-time motion detection and decision classification algorithm. This flowchart displays the major motion detection method and WiFi control method.

shows the workflow of motion detection. We calculate the 3-axis acceleration values $x(t)$, $y(t)$ and $z(t)$ from the accelerometer sensor, orientation sensor and magnetic sensor. The direction of motion, calculated from the trajectory record, helps to calculate moving speed. With the user's recent position information, we can calculate the current speed of motion. The speed, location records and acceleration values are provided to the motion detection component.

From Fig. 9, we use the user's current motion to determine the WiFi scan frequency and whether to turn on/off the Wifi adapter. The values of the 3-axis acceleration and history location records are used as input as shown in Fig. 9, to determine the user's current activities. There are five types of activities we need to recognize: standing, running, walking, going upstairs and going downstairs. For example, when the user step frequency (max amplitude's frequency) is around 0.7–3.0 Hz, the direct current value of $z(t)$ is above 10 and median value of $z(t)$ is above 0.5, there is a high probability that the tracked user is going upstairs. When $SMA < 1.5, x(t) < 0.6, y(t) < 0.6$, and $step\_frequency < 0.7$ Hz, there is a high probability that the user is standing still. For the standing activity, we do not need to record location records, and we can turn off the WiFi adapter. When the user is going upstairs, the median value of the accelerometer $z(t)$ will be above 0.5 (downstairs below -0.5). When a person is walking on a single floor, the value of the accelerometer $z(t)$ will not cross the threshold 0.5 and $\overline{x(t)} + \overline{y(t)}$ will cross 0.8. The accuracy of our detection algorithm for determining when a user is going upstairs is 72%. We can reduce the WiFi scan rate or turn off the WiFi when the user is going up or down, because the accelerometer can be used to calculate the vertical movement distance with high accuracy. When walking, the scan frequency is $1/\Delta t_s$, which is calculated using Eq. (12).

5. Performance evaluation

To assess the impact of the different power management strategies and data transfer protocols on energy power consumption and

Table 2
Accuracy measures for the proposed framework, kNN, and kernel-based approaches.

Alg.	Mean error (m)	Variance (m)	Complexity
Proposed	3.27	6.67	Moderate
kNN	2.74	8.53	Moderate
Kernel-based	2.31	4.57	High

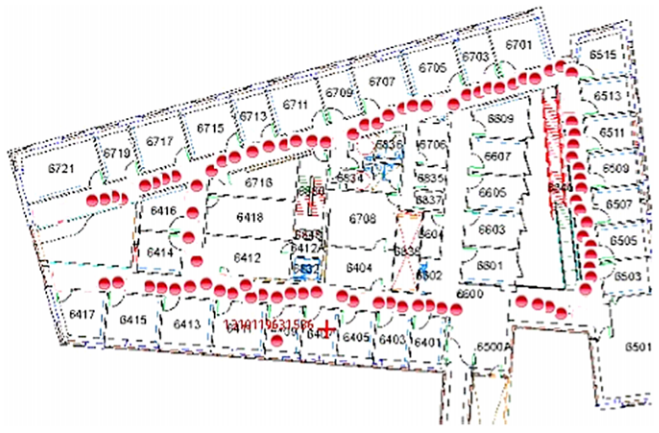


Fig. 10. One trajectory traversed by user for one day.

the accuracy requirement, we collected datasets from 3 people for one week. In the evaluation, we considered power models for (1) different localization technologies strategies (GSM, WiFi), (2) data transmission over GSM and WiFi and (3) CPU load.

5.1. Data collection

The system was tested in a building on our campus. A total of 2440 APs were detected over 9 floors, and RSS readings at 952 fingerprint points were collected. Our experiments were performed using an HTC G1 phone, which runs Android 1.6 with a 1150 mAh battery and only uses the WiFi interface. We recorded 3 people's daily trajectory information including position, motion, battery usage and timestamp. When the system predicted that a user's motion detection activity changed, half the time it presented the current motion prediction and asked them to verify or correct the prediction. We collected information for a week to validate the energy consumption of the approach with a guaranteed accuracy of about 4.5 m at the 85th percentile. Our participants were told to use the phone as they normally would: *e.g.*, calls, text, music, apps, web.

As our location algorithm (described in Section 3.1) is running on the mobile phone, it has high accuracy and short response time. We had each user stop at 20 known points in the building, and we calculated the error distance between ground truth and the point the algorithm reported. Table 2 compares the mean location accuracy with other systems, such as RADAR (using a *k*-Nearest Neighbor approach) [9,32] and a kernel-based framework [33].

The location algorithm is controlled by our energy control module. The location data is buffered in the smart phone, and is transmitted to the server as necessary using our data upload strategy.

5.2. Results

5.2.1. Location records

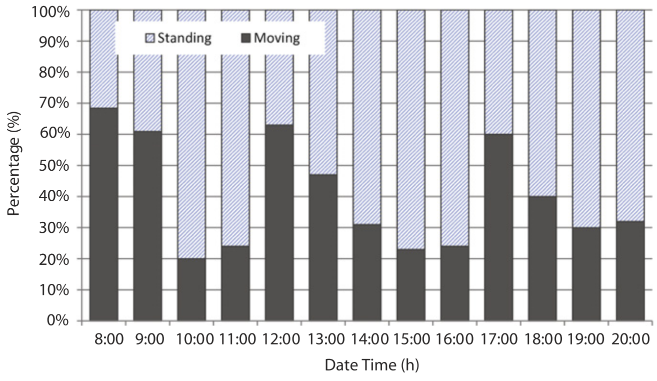
The framework keeps track of the user's location and motion, and updates a remote server in a timely manner. Fig. 10 shows a trace (red dots) captured by the framework of one participant for a single floor. During the time frame of the trace, the framework continues to run and update the server. We scan for available networks to detect if the user has left the building. Once detected, the framework scan frequency is reduced and the server is no longer updated, until the user is detected again.

5.2.2. Motion recognition accuracy

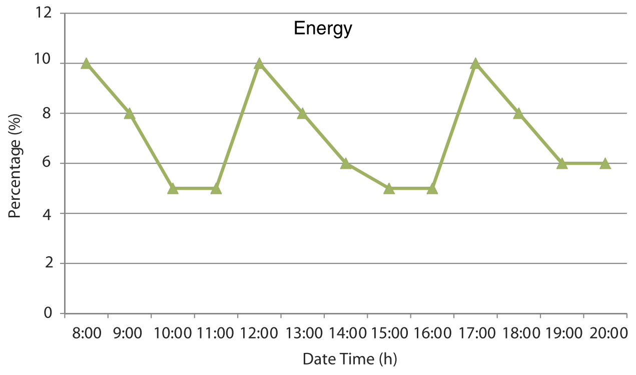
First, we examined the overall motion recognition accuracy for our users. Recognition accuracy is the ratio of the number of correctly recognized states to the total number of states. Fig. 11(a) shows a record of our participants from 8:00 to 20:00. The percentage of two activities, standing and moving, is summarized as columns for each hour. From 10:00 to 11:00 and 15:00 to 16:00, the participants are in a stationary state. At the same time, it consumes less energy than the other hours as shown in Fig. 11(b). From Fig. 11, we can see that more energy is consumed when there is more movement. The average motion recognition accuracy across all 3 users is 90% with a standard deviation of 2.53%. Our recognition accuracy varies slightly across users, suggesting that the recognition algorithm can be applied to general users, and not just the ones we used in our study.

5.2.3. Tracking accuracy

The use of energy saving management strategies potentially reduces the accuracy of the location tracking. In particular, when



(a) Activities percentage over time.



(b) Energy consumption over time.

Fig. 11. Standing motion for one user for one day.

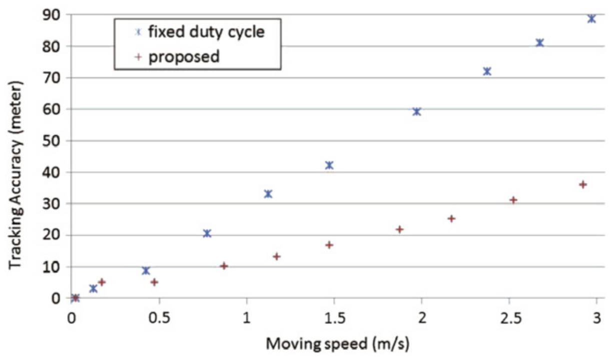


Fig. 12. Relationship between movement speed and tracking accuracy.

Table 3
Analysis of the amount of data captured.

User	Continuous tracking	Proposed tracking	Redundant data
A	1680	692	58.81%
B	1200	363	69.75%
C	1560	479	69.29%

Table 4
Evaluation of our framework.

User	In school time per day (h)	Device lifetime per day (h)	Motion recognition accuracy
A	12.2	13.7	88.24%
B	11.8	13.6	86.71%
C	11.1	14.1	90.82%

the smart phone is low on power, the sampling rate cannot be increased too much. We compare the accuracy of our framework with a system that continuously tracks location. The continuous tracking system does not use our energy saving strategies and performs a fixed WiFi scan every 30 s. Our framework consumes less energy and, as shown in Fig. 12, it has a reduced error for different moving speeds.

Our proposed tracking method does not record as much position data as a continuous tracking approach. People who are not moving do not need to be tracked. Also, due to weak WiFi signals in stairwells, when a person is going upstairs or downstairs, they often cannot be tracked accurately. So for those situations, we can turn off the WiFi adapter and stop tracking. By turning off the WiFi adapter, our method collects between 59% and 70% less data than the continuous approach, based on our experimental results. The details of this comparison work are shown in Table 3. Not only is energy saved by turning off the WiFi adapter, but additional energy is saved as the framework does not need to update the remote server when the user is stationary.

5.2.4. *Energy saving cost*

As an accelerometer is used to detect user's activities and calculate the distance traveled, it will also consume extra energy. However the energy spent on using the accelerometer is much less than the WiFi adapter, even if the accelerometer is always on [24]. In our framework, the accelerometer's delay mode is about 10–15 Hz and the energy consumption per minute is nearly 0 [24]. The minimal energy consumed is from the extra CPU usage.

5.2.5. *Device lifetime*

Table 4 shows the performance results for our framework. We asked users not to charge their phone until the battery was under 5%. The three participants are all graduate students, who usually stay in the building from 9am 10pm. Even with users performing other activities on the phone, the framework allowed the smart phone battery to last about 14 h. This is an improvement of 4 h, over the lifetime using the continuously sensing approach with 30 s updates (10 h with similar phone usage). Our framework can identify motion states with accuracy between 86% and 91%. Fig. 11 indicates that one of our users B sat for 70% of the time during working hours. If that result generalizes, and for many office employees it should, using the motion detection strategy can greatly extend mobile device lifetime. Note that the smart phone lifetime will differ for different users. For example, a phone whose owner is stationary most of the day will experience a longer lifetime than one whose owner is in motion most of the day.

6. Conclusion and future work

Supporting applications that require continuous location sensing on mobile phones is very challenging [34]. In this paper, we present the design, implementation and evaluation of an energy-efficient indoor tracking framework. The most important component for our framework is the sensor management module that monitors user activity and reduces energy consumption by shutting down unnecessary sensors and adaptively using WiFi. Our framework achieves around 14 h of lifetime, which should cover most people's working day. In addition, it achieves this lifetime in the presence of regular user phone activity, meaning that the framework has little impact on the user experience with their smartphone. While the tracking accuracy can be dynamically configured by different applications, we have validated that with an accuracy of 4.75 m, our framework is energy efficient. In future work, we plan to further improve the energy-efficiency by leveraging additional energy-efficient sensors to detect users' activities. We also plan on generalizing our sensor management approach to many other sensors, and apply this to more complex sensing applications.

Acknowledgments

The work is partly supported by NSFC (No. 61003220), Fundamental Research Funds for the Central Universities (HUST: 2010QN051), RFDP (No. 20090142120025) and Natural Science Foundation of Hubei Province of China (No. 2010CDB02302).

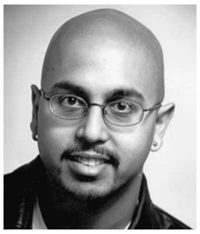
References

[1] Y. Chang, Y. Chu, C. Chen, T. Wang, Mobile computing for indoor wayfinding based on bluetooth sensors for individuals with cognitive impairments, in: Proceedings of the 3rd International Symposium on Wireless Pervasive Computing, IEEE, 2008, pp. 623–627.
[2] A. Liu, H. Hile, G. Borriello, P. Brown, M. Harniss, H. Kautz, K. Johnson, Customizing directions in an automated wayfinding system for individuals with cognitive impairment, in: Proceedings of the 11th International ACM SIGACCESS Conference on Computers and Accessibility, ACM, 2009, pp. 27–34.
[3] C. Yu, D. Yao, X. Li, Y. Zhang, L.T. Yang, N. Xiong, H. Jin, Location-aware private service discovery in pervasive computing environment, Inform. Sci. 230 (1) (2013) 78–93.
[4] J. Ng, Ubiquitous healthcare localisation schemes, in: Proceedings of 7th International Workshop on Enterprise Networking and Computing in Healthcare Industry, IEEE, 2005, pp. 156–161.
[5] J. Cadman, Deploying commercial location-aware systems, in: Proceedings of the 2003 Workshop on Location-Aware Computing, 2003, pp. 4–6.
[6] A. Harter, A. Hopper, P. Steggles, A. Ward, P. Webster, The anatomy of a context-aware application, Wirel. Netw. 8 (2) (2002) 187–197.
[7] Z. Zhuang, K.-H. Kim, J.P. Singh, Improving energy efficiency of location sensing on smartphones, in: Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services, ACM, 2010, pp. 315–330.
[8] A. LaMarca, Y. Chawathe, S. Consolvo, J. Hightower, I. Smith, J. Scott, T. Sohn, J. Howard, J. Hughes, F. Potter, et al., Place lab: device positioning using radio beacons in the wild, Pervasive Comput. (2005) 301–306.

[9] P. Bahl, V. Padmanabhan, Radar: an in-building rf-based user location and tracking system, in: Proceedings of the 19th IEEE Conference on Computer Communications, Vol. 2, IEEE, 2000, pp. 775–784.
[10] G. Sun, J. Chen, W. Guo, K. Liu, Signal processing techniques in network-aided positioning: a survey of state-of-the-art positioning designs, IEEE Signal Process. Mag. 22 (4) (2005) 12–23.
[11] M. Azizyan, I. Constandache, R. Roy Choudhury, Surroundsense: mobile phone localization via ambience fingerprinting, in: Proceedings of the 15th Annual International Conference on Mobile Computing and Networking, ACM, 2009, pp. 261–272.
[12] K. Lin, M. Chen, S. Zeadally, J.J. Rodrigues, Balancing energy consumption with mobile agents in wireless sensor networks, Future Gener. Comput. Syst. 28 (2) (2012) 446–456.
[13] N. Fernando, S.W. Loke, W. Rahayu, Mobile cloud computing: a survey, Future Gener. Comput. Syst. 29 (1) (2013) 84–106.
[14] N. Balasubramanian, A. Balasubramanian, A. Venkataramani, Energy consumption in mobile phones: a measurement study and implications for network applications, in: Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference, ACM, 2009, pp. 280–293.
[15] M. Kjaergaard, J. Langdal, T. Godsk, T. Toftkjær, Entracked: energy-efficient robust position tracking for mobile devices, in: Proceedings of the 7th International Conference on Mobile Systems, Applications, and Services, ACM, 2009, pp. 221–234.
[16] T. Farrell, R. Cheng, K. Rothermel, Energy-efficient monitoring of mobile objects with uncertainty-aware tolerances, in: Proceedings of the 11th IEEE International Conference on Database Engineering and Applications Symposium, IEEE, 2007, pp. 129–140.
[17] J. Paek, J. Kim, R. Govindan, Energy-efficient rate-adaptive gps-based positioning for smartphones, in: Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services, ACM, 2010, pp. 299–314.
[18] I. Constandache, S. Gaonkar, M. Sayler, R. Choudhury, L. Cox, EnLoc: energy-efficient localization for mobile phones, in: Proceedings of the 28th IEEE Conference on Computer Communications, IEEE, 2009, pp. 2716–2720.
[19] S. Gaonkar, J. Li, R. Choudhury, L. Cox, A. Schmidt, Micro-blog: sharing and querying content through mobile phones and social participation, in: Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services, ACM, 2008, pp. 174–186.
[20] K. Lin, A. Kansal, D. Lymberopoulos, F. Zhao, Energy-accuracy trade-off for continuous mobile device location, in: Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services, ACM, 2010, pp. 285–298.
[21] M. Haridasan, I. Mohomed, D. Terry, C. Thekkath, L. Zhang, Startrack next generation: a scalable infrastructure for track-based applications, in: Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation, USENIX Association, 2010, pp. 1–6.
[22] M.B. Kjærgaard, S. Bhattacharya, H. Blunck, P. Nurmi, Energy-efficient trajectory tracking for mobile devices, in: Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services, ACM, 2011, pp. 307–320.
[23] I. Constandache, X. Bao, M. Azizyan, R. Choudhury, Did you see Bob?: Human localization using mobile phones, in: Proceedings of the 16th Annual International Conference on Mobile Computing and Networking, ACM, 2010, pp. 149–160.
[24] D. Kim, Y. Kim, D. Estrin, M. Srivastava, Sensloc: sensing everyday places and paths using less energy, in: Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems, ACM, 2010, pp. 43–56.
[25] I. Shafer, M. Chang, Movement detection for power-efficient smartphone wlan localization, in: Proceedings of the 13th ACM International conference on Modeling, Analysis, and Simulation of Wireless and Mobile Systems, ACM, 2010, pp. 81–90.
[26] J. Ledlie, J. Park, D. Curtis, A. Cavalcante, L. Camara, A. Costa, R. Vieira, Mole: a scalable, user-generated wifi positioning engine, J. Locat. Based Serv. 6 (2) (2012) 55–80.
[27] F. Dellaert, D. Fox, W. Burgard, S. Thrun, Monte Carlo localization for mobile robots, in: Proceedings of IEEE International Conference on Robotics and Automation, Vol. 2, IEEE, 1999, pp. 1322–1328.
[28] A. Rahmati, L. Zhong, Context-for-wireless: context-sensitive energy-efficient wireless data transfer, in: Proceedings of the 5th International Conference on Mobile Systems, Applications and Services, ACM, 2007, pp. 165–178.
[29] Android Market. URL http://www.android.com/market.
[30] M. Mathie, Monitoring and interpreting human movement patterns using a triaxial accelerometer, Ph.D. Thesis, The University of New South Wales, 2003.
[31] W. Chung, A. Purwar, A. Sharma, Frequency domain approach for activity classification using accelerometer, in: Proceedings of the 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, 2008, pp. 1120–1123.
[32] Y. Gu, A. Lo, I. Niemegeers, A survey of indoor positioning systems for wireless personal networks, IEEE Commun. Surv. Tutor. 11 (1) (2009) 13–32.
[33] A. Kushki, K. Plataniotis, A. Venetsanopoulos, Kernel-based positioning in wireless local area networks, IEEE Trans. Mob. Comput. 6 (6) (2007) 689–705.
[34] H. Lu, J. Yang, Z. Liu, N. Lane, T. Choudhury, A. Campbell, The jigsaw continuous sensing engine for mobile phone applications, in: Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems, ACM, 2010, pp. 71–84.

Dezhong Yao is a Ph.D. student at the Services Computing Technology and System Lab, Huazhong University of Science and Technology. His research interests include wireless networks, ubiquitous computing, context-awareness and data mining.

Chen Yu received the B.S. degree in mathematics and the M.S. degree in computer science from Wuhan University in 1998 and 2002, respectively. He received the Ph.D. degree in information science from Tohoku University in 2005. He had worked as a Japan Science and Technology Agency postdoctoral researcher in Japan Advanced Institute of Science and Technology from 2005 to 2006. From 2006, he had worked as a Japan Society for the Promotion of Science postdoctoral fellow in Japan Advanced Institute of Science and Technology. Since 2008, he has been with the School of Computer Science and Technology, Huazhong University of Science and Technology, China, where he is an associate professor and a special research fellow, working in the areas of Wireless Sensor Networks, Ubiquitous Computing, and Green Communications. He has received the Best Paper Award in the IEEE International Conference on Communication (ICC '05) and the nominated Best Paper Award in the Proceedings of the 11th IEEE International Symposium on Distributed Simulation and Real-Time Applications (DS-RT '07).



Anind K. Dey is Associate Professor in the Human-Computer Interaction (HCI) Institute at Carnegie Mellon University. He received a Bachelors of Applied Science in Computer Engineering from Simon Fraser University in Burnaby, Canada in 1993. He earned his Masters of Science in AeroSpace Engineering from Georgia Tech in 1995. He earned his 2nd Masters of Science and a Ph.D. in Computer Science at Georgia Tech in 2000. He was a member of the Future Computing Environments research group in the College of Computing at Georgia Tech. He was a Senior Researcher at Intel Research Berkeley from 2001 to 2004, where his title was Ubicomp Software Architect. At the same time, he was an Adjunct Assistant Professor in the EECS Department at UC Berkeley, where he was a member of GUIR, the Group for User Interface Research. His research interests are feedback and control in ubiquitous computing, context-aware computing, toolkits and end-user programming environments, sensor-rich environments, information overload, ambient displays, privacy, human-computer interaction, machine learning.

Christian Koehler is a Ph.D. student in the Electrical and Computer Engineering Department at Carnegie Mellon University, Pittsburgh, advised by Anind Dey. My research interest lies in using applied Machine Learning, especially Location Prediction, to find solutions for the ever growing resource consumption problem of our society.

Geyong Min is a Professor in Computer Science in the Department of Computing at the University of Bradford. He received the Ph.D. degree in Computing Science from the University of Glasgow, UK, in 2003, and the B.Sc. degree in Computer Science from Huazhong University of Science and Technology, China, in 1995. He joined the University of Bradford as a Lecturer in 2002, became a Senior Lecturer in 2005 and a Reader in 2007. He was promoted to a Professor in 2012.

Laurence T. Yang graduated from Tsinghua University, China and got his Ph.D. in Computer Science from University of Victoria, Canada. He joined St. Francis Xavier University in 1999. His current research includes parallel and distributed computing, embedded and ubiquitous/pervasive computing. He has published many papers in various refereed journals, conference proceedings and book chapters in these areas (IEEE Journal on Selected Areas in Communications, IEEE Transactions on System, Man and Cybernetics, IEEE Transactions on Very Large Scale Integration Systems, IEEE Transactions on Industrial Informatics, IEEE Transactions on Information Technology in Biomedicine, IEEE Transactions on Parallel and Distributed Systems, IEEE Transactions on Circuit and Systems, IEEE Transactions on Service Computing, ACM Transactions on Embedded Computing Systems, IEEE Systems Journal, ACM Transactions on Autonomous and Adaptive Systems, IEEE Transactions on Vehicular Technology, ACM/Springer Journal on Mobile Networks and Applications, IEEE Intelligent Systems, Journal of Parallel and Distributed Computing, etc). He is also in the steering committee of IEEE/ACM Supercomputing conference series, and the National Resource Allocation Committee (NRAC) of Compute Canada.

Hai Jin is a Professor of Computer Science and Engineering at the Huazhong University of Science and Technology (HUST) in China. He is now the Dean of School of Computer Science and Technology at HUST. He received his Ph.D. in computer engineering from HUST in 1994. In 1996, he was awarded German Academic Exchange Service (DAAD) fellowship for visiting the Technical University of Chemnitz in Germany. He worked for the University of Hong Kong between 1998 and 2000 and participated in the HKU Cluster project. He worked as a visiting scholar at the University of Southern California between 1999 and 2000. He is the chief scientist of the largest grid computing project, ChinaGrid, in China, and he is the director of Key Lab of Service Computing Technology and System, MOE (Ministry of Education). Also, he is the chief scientist of National 973 Basic Research Program, Basic Theory and Methodology of Virtualization Technology for Computing System. Dr. Jin is a senior member of IEEE and a member of ACM. He is the associated editor of International Journal of Computer and Applications, International Journal of Grid and Utility Computing, International Journal of Web and Grid Services, Journal of Computer Science and Technology. He is IASTED technical committee member on Parallel & Distributed Computing and Systems. He is the steering committee chair of International Conference on Grid and Pervasive Computing (GPC). He is the steering committee member of IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid) and International Conference on Grid and Utility Computing, and served as program chair of GCC '04, SCC '05, HSI '05, NPC '05, and UISW '05, program vice-chair of the CCGrid '01, PDCAT '03, NPC '04, EUC '05, e-Science '05, AINA '06, and CCGrid '06. He served as conference chairs for International Workshop on Internet Computing and E-Commerce from 2001 to 2003. He also served as program committee for more than 200 international conferences/workshops. He has co-authored 10 books and published over 200 research papers. His research interests include cluster computing and grid computing, peer-to-peer computing, network storage, network security, and virtualization technology.