Leveraging GPS-Less Sensing Scheduling for Green Mobile Crowd Sensing

Xiang Sheng, *Student Member, IEEE*, Jian Tang, *Senior Member, IEEE*, Xuejie Xiao, and Guoliang Xue, *Fellow, IEEE*

***Abstract*—In this paper, we consider leveraging GPS-less energy-efficient sensing scheduling for mobile crowd sensing. We present a probabilistic model for sensing coverage without accurate location information (provided by GPS), based on which we formally define the Energy-constrained Maximum Coverage Sensing Scheduling (E-MCSS) problem for maximum coverage and the Fair Maximum Coverage Sensing Scheduling (F-MCSS) problem for fairness. Assuming that moving trajectories of mobile users are known beforehand, we present a $(1 - 1/e)$-approximation algorithm and a $1/2$-approximation algorithm to solve the E-MCSS and F-MCSS problems in polynomial time, respectively, which can serve as benchmarks for performance evaluation. Under realistic assumptions, we present a GPS-less energy-efficient protocol for sensing scheduling based on the proposed algorithms. We developed an Android-based mobile crowd sensing system, on which we implemented the proposed protocol. Simulation results and experimental results (from a field test) are presented to validate and justify effectiveness of the proposed algorithms and protocol.**

***Index Terms*—Collaborative sensing, energy-efficiency, mobile crowd sensing, scheduling.**

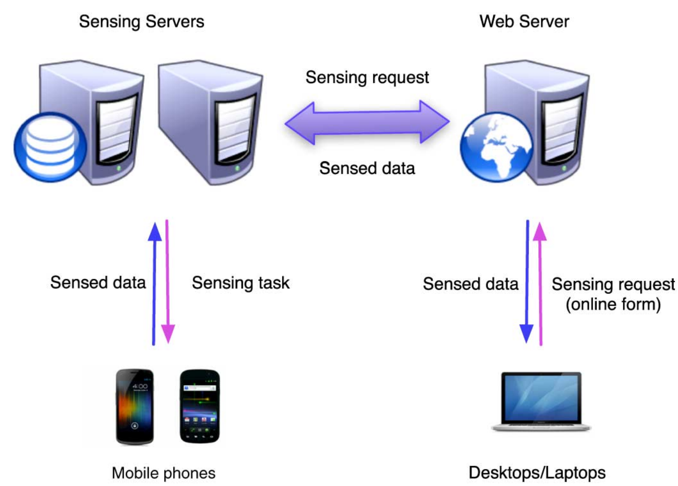


Fig. 1. Mobile crowd sensing system.

I. Introduction

MOBILE PHONES have evolved as key electronic devices for communications, computing, and entertainment, and have become an important part of people's daily lives. Most of current mobile phones (such as iPhone 5/5S and Google Nexus 4/5) are also equipped with a rich set of embedded sensors such as camera, GPS, WiFi/3G/4G interfaces, accelerometer, digital compass, gyroscope, microphone, and so on [12]. Moreover, external sensors (such as Fitbit [9] and Sensordrone [24]) can also be connected to a mobile phone via its Bluetooth interface. These sensors can enable attractive sensing applications in various domains [12] such as environmental monitoring, social network, healthcare, transportation, and safety.

There are primarily two mobile crowd sensing paradigms [12]: 1) *participatory sensing* and 2) *opportunistic sensing*. In participatory sensing, users actively engage in sensing activities (e.g., take a picture and record a video) by manually determining when, where, what, and how to sense. In opportunistic sensing, sensing activities are fully automated without user involvement (e.g., scan WiFi signals and record noise samples). Both of them have been studied by a few recent works [12], most of which, however, presented a mobile crowd sensing system for a particular application (e.g., PEIR [18] and Ear-Phone [22]).

We consider a mobile crowd sensing system consisting of web servers, sensing servers, and mobile phones (that run the sensing application), as illustrated in Fig. 1. Multiple sensing servers (which could be physical servers or virtual-machine-powered servers) can be deployed to handle sensing requests from different locations. This system provides sensing services, in which mobile phone users can be not only service users but also service providers. When a person initiates a sensing request through an online form in a web server from either a mobile phone or a regular computer (desktop/laptop), the request will be forwarded to a sensing server which will then push the request to a subset of mobile phones that happen to be in the area of interest (i.e., sensing crowd). The corresponding sensing task will then be fulfilled by mobile phones (whose users agree to participate), and sensed data will be collected and uploaded to the sensing server. Finally, the sensed data will be sent back to the requester via the web server.

Manuscript received January 26, 2014; revised April 25, 2014; accepted June 02, 2014. Date of publication July 01, 2014; date of current version August 01, 2014. This work was supported by NSF under Grant 1218203 and Grant 1217611.

X. Sheng, J. Tang, and X. Xiao are with the Department of Electrical Engineering and Computer Science, Syracuse University, Syracuse, NY 13244 USA (e-mail: xsheng@syr.edu; jtang02@syr.edu; xxiao04@syr.edu).

G. Xue is with the School of Computing, Informatics, and Decision Systems Engineering, Arizona State University, Tempe, AZ 85287 USA (e-mail: xue@asu.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JIOT.2014.2334271

2327-4662 © 2014 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

Although mobile crowd sensing applications are very attractive, performing sensing tasks may consume a significant amount of energy of a mobile phone. Moreover, most mobile crowd sensing applications are location-dependent, which may require location information to be reported along with sensed data. If energy-hungry GPS is turned ON during the whole sensing procedure, the battery may be drained very quickly. Hence, without carefully managing very limited energy resources, a mobile phone may end up with running out of its battery after performing a few sensing tasks.

There is a large space for energy savings on a mobile phone. However, fundamental energy-efficient resource management problems have not been carefully studied in the context of mobile crowd sensing.

In this paper, we aim to design scheduling algorithms and a protocol for mobile crowd sensing without accurate locations (provided by GPS) with the objective of achieving a specified coverage requirement with limited energy consumption. Our contributions are summarized in the following.

1) We present a probabilistic model for sensing coverage without accurate location information, based on which we formally define the Energy-constrained Maximum Coverage Sensing Scheduling (E-MCSS) problem for maximum coverage and the Fair Maximum Coverage Sensing Scheduling (F-MCSS) problem for addressing fairness on individual energy usages.
2) Assuming moving trajectories of mobile users are known in advance, we present a $(1 - 1/e)$-approximation algorithm and a $1/2$-approximation algorithm to solve the E-MCSS and F-MCSS problems in polynomial time, respectively. Even though this assumption might not be realistic, the solutions given by these theoretically sound algorithms can serve as benchmarks for performance evaluation.
3) Under realistic assumptions, we present a GPS-less energy-efficient protocol for sensing scheduling based on the proposed approximation algorithms.
4) We developed an Android-based mobile crowd sensing system, on which we implemented the proposed protocol. We present simulation results based on real location data (collected from the Google Map) as well as experimental results from a field test on Syracuse University's campus to validate and justify effectiveness of the proposed algorithms and protocol.

To the best of our knowledge, we are the first to present theoretically well-founded and practically efficient mathematical model, algorithms, and protocol for coverage (without accurate locations) and energy-efficient sensing scheduling in the context of mobile crowd sensing.

II. ENERGY-EFFICIENT GPS-LESS SENSING SCHEDULING

A. Problem Definition

Sensing scheduling is only considered for opportunistic sensing applications since mobile users usually control sensing activities manually in participatory sensing applications.

The movement of a mobile user k can be characterized using a trajectory Γ_k that is a set of 3-tuples (k, t_k, loc_k). Each tuple gives the location of user k at time t_k. The more 3-tuples there are in the trajectory, the more accurately it can characterize user's movement. Usually a mobile user only carries one phone; therefore, we use "mobile phone" and "user" interchangeably.

Ideally, a mobile phone should send sensor readings along with the corresponding locations obtained from its GPS to a sensing server. However, it is well known that GPS is very energy-hungry [13] (a GPS device usually consumes much more energy than other sensors). Keeping GPS ON during the whole sensing procedure is not feasible since it may drain the battery quickly. Other approaches, such as Google's location service [20], can also be used to obtain location information from a remote Google server by calling an Android system API, which usually consumes much less power [3], but provides less accuracy compared to GPS. We consider a *GPS-less* system in which each mobile phone uses certain location service (such as Google) to obtain location information without turning ON its GPS.

Every 3-tuple (k, t_k, loc_k) in a trajectory can be imagined as a *virtual sensor*. If trajectories of all users are given, then we can have a large network of *virtual sensors* by combining all 3-tuples in trajectories. A sensing schedule S is a collection of virtual sensor sets, i.e., $S = \bigcup_{k=1}^{K} S_k$, where $S_k \subseteq \Gamma_k$. Selecting a virtual sensor (k, t_k, loc_k) into the schedule essentially means that mobile phone k is scheduled to sense at the location loc_k and time t_k. $|S_k| = |S \bigcap \Gamma_k|$ gives the number of times mobile phone of user k senses.

Sensing targets are assumed to be a set V of points. If the accurate location of each virtual sensor is known, then it is easy to figure out whether a target point $v_j \in V$ can be covered by a virtual sensor s_i. However, since a GPS-less location approach usually cannot provide precise locations, we present a probabilistic model to estimate the probability that a virtual sensor s_i covers a target point v_j with location errors (P_{ij}) and the probability that a target point v_j is covered by a given sensing schedule (P_j), which will be discussed in the next section. Since coverage quality and energy efficiency are the primary design goals, we consider a problem of maximizing sensing coverage subject to sensing energy consumption constraints. We formally define the optimization problem in the following.

Definition 1 (E-MCSS): Given a set V of J target points, K mobile phone users, a sensing deadline T, and the moving trajectory Γ_k of each user $1 \leq k \leq K$ before the deadline, the *E-MCSS* problem seeks a sensing schedule $S = \bigcup_{k=1}^{K} S_k$, where $S_k \subseteq \Gamma_k$ and $1 \leq k \leq K$, such that the total coverage probability $\sum_{j=1}^{J} P_j$ is maximized while the constraint that the total number of sensing times $\sum_{k=1}^{K} |S_k| \leq B$ before the deadline T, where B is a given threshold (which we call the *sensing budget*) is satisfied.

However, bounding the total number of sensing times only may lead to unfair utilizations of mobile phones: some users' phones are heavily used for sensing and other users' phones are lightly utilized or not utilized at all. Therefore, we try to improve fairness by considering those sensing schedules in which each user's sensing time is bounded. We call such schedules *fair* sensing schedules. We also study the *F-MCSS* problem which seeks a fair sensing schedule $S = \bigcup_{k=1}^{K} S_k$, where $S_k \subseteq \Gamma_k$ and $|S_k| \leq B_k$, $1 \leq k \leq K$ (B_k is the sensing budget of mobile user k). We will discuss how to set B_k and B in practice in Section IV.

B. Sensing Coverage Model

In this section, we present a probabilistic model for sensing coverage with location errors.

The Google's location service [20] provides a rough estimation for the location of a mobile phone, which is given by a 3-tuple, (x, y, R), where x and y are latitude and longitude of the estimated location, respectively, and R is the accuracy radius which can be obtained from the MaxMind Accuracy Radius database [16]. With 50% probability, the phone is located inside the *location disk* with origin at (x, y) and a radius of R. The actual location is assumed to follow a two-dimensional (2-D) normal distribution with correlation $\rho_{xy} = 0$.

Suppose that a virtual sensor s_i has a location of (x_i, y_i, R_i) given by the Google's location service. Note that this location may not be accurate and it is called *reported location* in the following. We present a method to calculate P_{ij}, i.e., the probability that a particular target point $v_j \in V$ can be covered by this virtual sensor (i.e, if a mobile phone senses at this location, what is the probability that this target point can be covered). First, it is easy to calculate the probability that a mobile user actually shows up at a specific location. If the sensing range is R', the probability that a target v_i can be covered equals the probability that a virtual sensor is located inside the corresponding *target disk* with origin at v_i and a radius of R', which is given by the following equation:

$$P_{ij} = \int_{\phi_1}^{\phi_2} \int_{r_1}^{r_2} P(r) r \mathrm{d}r \mathrm{d}\phi. \tag{1}$$

Since the distance between the actual location and the reported location r follows a 2-D normal distribution with $\sigma_x = \sigma_y = \sigma$, the probability that the actual location is r away from the reported location can be given by the following equation:

$$P(r) = \frac{1}{2\pi\sigma^2} \exp\left\{-\frac{r^2}{2\sigma^2}\right\}. \tag{2}$$

As shown in Fig. 2, ϕ is the angle between the line with the reported and actual locations and the line with the reported location and the target point. $[r_1, r_2]$ and $[\phi_1, \phi_2]$ are the integration intervals for r and ϕ, respectively. As mentioned above, the probability that a virtual sensor s_i is actually inside the location disk with origin at the reported location and a radius of R_i is 50%; therefore, we can obtain σ

$$\sigma \approx 1.5 R_i. \tag{3}$$

As shown in (2), $P(r)$ is independent of ϕ, so (1) can be simplified to

$$P_{ij} = \int_{r_1}^{r_2} P(r)(\phi_2(r) - \phi_1(r)) r \mathrm{d}r. \tag{4}$$

We need to consider two cases for the distance d_{ij} between virtual sensor s_i and the target point v_j: 1) $d_{ij} \geq R'$ and 2) $d_{ij} < R'$, which are shown in Fig. 2.

For case 1: $d_{ij} \geq R'$, using the law of cosine, we can have

$$\phi_2(r) - \phi_1(r) = 2\cos^{-1}\left(\frac{r^2 + d_{ij}^2 - R'^2}{2 r d_{ij}}\right). \tag{5}$$

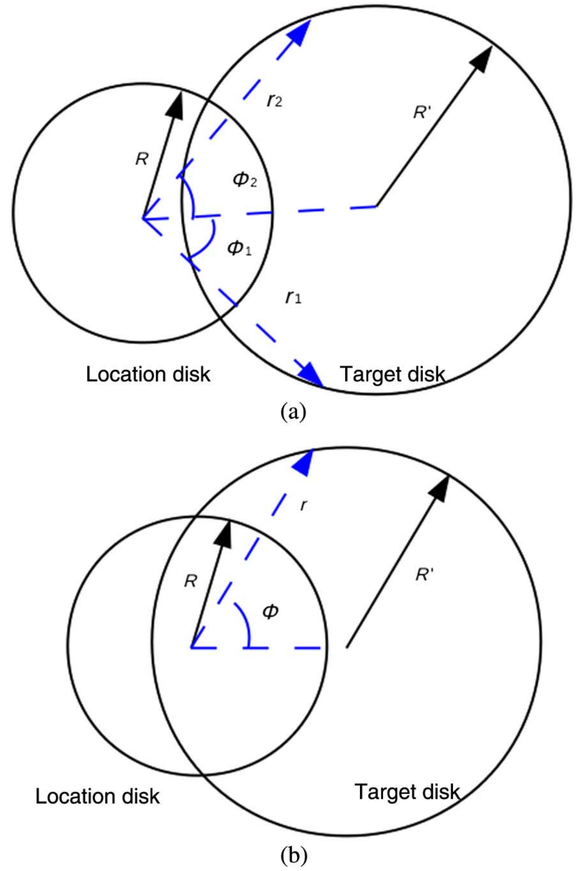


Fig. 2. Sensing coverage model. (a) Case 1: $d_{ij} \geq R'$. (b) Case 2: $d_{ij} < R'$.

By replacing $P(r)$ and $(\phi_2(r) - \phi_1(r))$ in (4) with (2) and (5), we have

$$P_{ij} = \int_{d_{ij}-R'}^{d_{ij}+R'} \frac{1}{\pi\sigma^2} \exp\left\{-\frac{r^2}{2\sigma^2}\right\} \times \cos^{-1}\left(\frac{r^2 + d_{ij}^2 - R'^2}{2 r d_{ij}}\right) r \mathrm{d}r. \tag{6}$$

For case 2: $d_{ij} < R'$, we have

$$\phi_2(r) - \phi_1(r) = \begin{cases} 2\pi, & \text{if } 0 \leq r \leq R' - d_{ij} \\ 2\cos^{-1}\left(\frac{r^2 + d_{ij}^2 - R'^2}{2 r d_{ij}}\right), & \text{if } r > R' - d_{ij}. \end{cases} \tag{7}$$

Similarly, we have

$$P_{ij} = \int_0^{R'-d_{ij}} \frac{1}{\sigma^2} \exp\left\{-\frac{r^2}{2\sigma^2}\right\} r \mathrm{d}r + \int_{R'-d_{ij}}^{d_{ij}+R'} \frac{1}{\pi\sigma^2} \exp\left\{-\frac{r^2}{2\sigma^2}\right\} \times \cos^{-1}\left(\frac{r^2 + d_{ij}^2 - R'^2}{2 r d_{ij}}\right) r \mathrm{d}r. \tag{8}$$

Since there does not exist the closed-form integration for the functions in (6) and (8), we use the numerical method to obtain approximate solutions. Note that the number of integration steps were set to $H = 100$ in the simulation and experiment.

For case 1: $d_{ij} \geq R'$, based on (6), we can have

$$\delta = \frac{2R'}{H} \tag{9}$$

$$r = (d_{ij} - R' + h\delta) \tag{10}$$

$$P_{ij} = \sum_{h=0}^{h=H} \left(\frac{r\delta}{\pi\sigma^2} \exp\left\{-\frac{r^2}{2\sigma^2}\right\} \times \cos^{-1}\left(\frac{d_{ij}^2 + r^2 - R'^2}{2d_{ij}r}\right)\right). \tag{11}$$

For case 2: $d_{ij} < R'$, based on (8), we can have

$$\delta = \frac{d_{ij} + R'}{H} \tag{12}$$

$$r = h\delta \tag{13}$$

$$P_{ij} = \sum_{h=0}^{h=\lfloor \frac{R'-d_{ij}}{\delta} \rfloor} \frac{r\delta}{\sigma^2} \exp\left\{-\frac{r^2}{2\sigma^2}\right\} \tag{14}$$

$$+ \sum_{h=\lceil \frac{R'-d_{ij}}{\delta} \rceil}^{h=H} \frac{r\delta}{\pi\sigma^2} \exp\left\{-\frac{r^2}{2\sigma^2}\right\} \times \cos^{-1}\left(\frac{d_{ij}^2 + r^2 - R'^2}{2d_{ij}r}\right). \tag{15}$$

So given a set of virtual sensor S, the probability that a target point j can be covered [by some virtual sensor(s) in S] is given as follows:

$$P_j = \left(1 - \prod_{s_i \in S} (1 - P_{ij})\right). \tag{16}$$

C. Proposed Approximation Algorithms

Assuming and knowing the trajectory of each mobile user in advance, we present constant factor approximation algorithms to solve the E-MCSS problem and the F-MCSS problem, respectively. It may be argued that this assumption is not realistic since it is hard to precisely predict how users will move in the future. However, the provably good solutions given by these algorithms can be used to show sensing coverage that can potentially be achieved by collaborative sensing and to serve as benchmarks for performance evaluation.

The E-MCSS problem can be formally formulated as follows:

$$\max \sum_{j=1}^{J} \left(1 - \prod_{s_i \in S} (1 - P_{ij})\right). \tag{17}$$

Subject to

$$|S| \leq B. \tag{18}$$

Basically, the problem is to maximize the total sensing coverage probability by selecting a subset S of virtual sensors with a cardinality no more than the given budget B. Mathematically, the E-MCSS problem has the same objective function as that of the budget server problem with a uniform cost (BSP-UC) studied in [30]. It has been shown in [30] that this objective function is a nonnegative, monotone, and submodular function. In a well-known work [19], Nemhauser *et al.* proved that, for such a problem, a simple greedy algorithm guarantees a solution with value at least $(1 - 1/e) > 0.63$ of the optimal. We present the greedy algorithm in the following, which is a $(1 - 1/e)$-approximation algorithm for the E-MCSS problem.

Algorithm 1. The greedy algorithm for the E-MCSS problem (E-MCSS-Greedy)

Step 1	$S_0 = \emptyset$;
Step 2	**for** $h = 1$ **to** B
	$s := \text{argmax}_{s' \in \Gamma \setminus S_{h-1}} f(S_{h-1} \bigcup \{s'\}) - f(S_{h-1})$;
	$S_h := S_{h-1} \bigcup \{s\}$;
	endfor
Step 3	**return** S_B;

In this algorithm, $f(\cdot)$ is the objective function, i.e., $f(S) = \sum_{j=1}^{J} \left(1 - \prod_{s_i \in S}(1 - P_{ij})\right)$. This algorithm starts with an empty set and adds a virtual sensor that maximizes the incremental objective value in each iteration. The running time of this algorithm is $O(B|\Gamma|)$, where $\Gamma = \bigcup_{k=1}^{K} \Gamma_k$ is the ground set of virtual sensors.

Next, we consider the F-MCSS problem and present the problem formulation in the following:

$$\max \sum_{j=1}^{J} \left(1 - \prod_{s_i \in S} (1 - P_{ij})\right).$$

Subject to

$$\left|S \bigcap \Gamma_k\right| \leq B_k, 1 \leq k \leq K. \tag{19}$$

We construct $\Omega = \{S_l : S_l \subseteq \Gamma, |S_l \bigcap \Gamma_k| \leq B_k, 1 \leq k \leq K\}$, which is a collection of subsets of the ground set Γ. We assume $\emptyset \in \Omega$. Next, we show that (Γ, Ω) is a matroid.

Definition 2 (Matroid [11]): A pair $(U, \mathcal{Z})$ consisting of a ground set U and a collection $\mathcal{Z}$ of subsets of U is a matroid:

1) if $X \in \mathcal{Z}$ and $Y \subset X$, then $Y \in \mathcal{Z}$;
2) for all $X, Y \in \mathcal{Z}$, if $|X| > |Y|$ then there exists some $x \in X \backslash Y$ such that $Y \bigcup \{x\} \in \mathcal{Z}$.

Lemma 1: (Γ, Ω) is a matroid.

Proof. Suppose that $S_{l_1} \in \Omega$. According to the definition of Ω, S_{l_1} satisfies the constraint $|S_{l_1} \bigcap \Gamma_k| \leq B_k, 1 \leq k \leq K$. And if $S_{l_2} \subset S_{l_1}$, then we have $|S_{l_2} \bigcap \Gamma_k| \leq |S_{l_1} \bigcap \Gamma_k| \leq B_k, 1 \leq k \leq K$. So $S_{l_2} \in \Omega$.

We prove that condition 2) is also satisfied by contradiction. Suppose that $S_{l_1} \in \Omega, S_{l_2} \in \Omega$, and $|S_{l_1}| > |S_{l_2}|$, but there does not even exist any element x such that $x \in S_{l_1} \setminus S_{l_2}$ and $S_{l_2} \bigcup \{x\} \in \Omega$. If this statement is not true, then $S_{l_2} \bigcup \{x_k\} > B_k, \forall x_k \in \{S_{l_1} \backslash S_{l_2}\} \bigcap \Gamma_k, 1 \leq k \leq K$. This means for any k, if $\{S_{l_1} \backslash S_{l_2}\} \bigcap \Gamma_k \neq \emptyset$, then $|S_{l_2} \bigcap \Gamma_k| = B_k$. So $|S_{l_1} \bigcap \Gamma_k| \leq |S_{l_2} \bigcap \Gamma_k|, \forall \{S_{l_1} \setminus S_{l_2}\} \bigcap \Gamma_k \neq \emptyset, 1 \leq k \leq K$. And since $\{S_{l_1} \backslash S_{l_2}\} \subset S_{l_1}$, and all the elements in $S_{l_1} \bigcap S_{l_2}$ are shared by both S_{l_1} and S_{l_2}, $|S_{l_1}| \leq |S_{l_2}|$, which is in contradiction to our assumption. This completes our proof. ■

The F-MCSS problem can be reformulated as follows:

$$\max_{S\in\Omega}\sum_{j=1}^{J}\left(1-\prod_{s_i\in S}(1-P_{ij})\right). \tag{20}$$

Therefore, the F-MCSS problem belongs to a class of problems of maximizing a submodular set function over a matroid [11]. We present a greedy algorithm to solve it and we have the following theorem.

Algorithm 2. The greedy algorithm for the F-MCSS problem (F-MCSS-Greedy)

Step 1 $S_0 := \emptyset$; $h := 1$;
Step 2 **while** $\exists s \in \Gamma \backslash S_{h-1}$ s.t. $S_{h-1}\bigcup\{s\} \in \Omega$
$s := \mathrm{argmax}_{s'\in S\backslash S_{h-1}} f(S_{h-1}\bigcup\{s'\}) - f(S_{h-1})$;
$S_h := S_{h-1}\bigcup\{s\}$;
$h := h+1$;
endwhile
Step 3 **return** S_h;

Theorem 1: Algorithm 2 is a $1/2$-approximation algorithm for the F-MCSS problem and has a time complexity of $O(N^2)$.

Proof. In [11], it is shown that a simple greedy algorithm gives a $1/2$ approximation for an optimization problem of maximizing a nondecreasing submodular set function over a matroid. The idea of Algorithm 2 is the same as the greedy algorithm presented in [11]; therefore, according to Lemma 1, Algorithm 2 is a $1/2$-approximation algorithm for the F-MCSS problem. The running time is $O(|\Gamma|^2 \cdot p(|\Gamma|))$, where $p(|\Gamma|)$ is the running time for testing whether $S_{h-1}\bigcup\{s\} \in \Omega$ or not. In our case, this testing can be easily done in constant time by maintaining a counter for each user and checking if its value exceeds the given budget.

D. Proposed Sensing Scheduling Protocol

In this section, we present a protocol which leverages the algorithms described above for sensing scheduling. As mentioned above, both algorithms need to know the trajectories of all mobile phones in the whole sensing period beforehand, which is not quite possible in practice. Therefore, we need to find a practical way to apply the proposed sensing scheduling algorithms.

Fig. 3 illustrates how the proposed protocol works. Each mobile phone periodically reports its location (obtained from the Google's location service) to a sensing server. Note that the location updating period was set to 20 s in our field test. The sensing server keeps track of recent locations for each mobile phone (currently, our system keeps 10 most recent locations for each phone). When a new sensing request (that specifies what to sense and area of interest) arrives, the sensing server will push the request to mobile phones that happen to be in the area of interest specified by the request and wait for confirmations from them. Then the sensing server will apply a sensing scheduling algorithm to schedule sensing activities of a set of mobile phones that confirm their willingness to participate, and instruct them to sense by sending them a schedule that specifies when to sense. Each phone will then use proper sensors to sense, encapsulate sensed data in an HTTP message and forward it to the sensing server. Any algorithm can be applied here for sensing scheduling. Of course, we employ the two algorithms presented above in our system. However, in order to use them, we need to predict the moving trajectory for each phone in the sensing period specified by the request, which is hard if the period is too long. In our protocol, we divide the whole sensing period into timeslots with the same duration (which was set to 200 s in the simulation and field test) and schedule sensing activities at the beginning of each timeslot. Specifically, during each timeslot, the sensing server performs the following actions:

1) uses a mobility prediction algorithm to predict the moving trajectory of each mobile phone;
2) feeds the predicted trajectories to a scheduling algorithm to obtain a schedule and informs each mobile phone;
3) collects the sensed data from mobile phones.

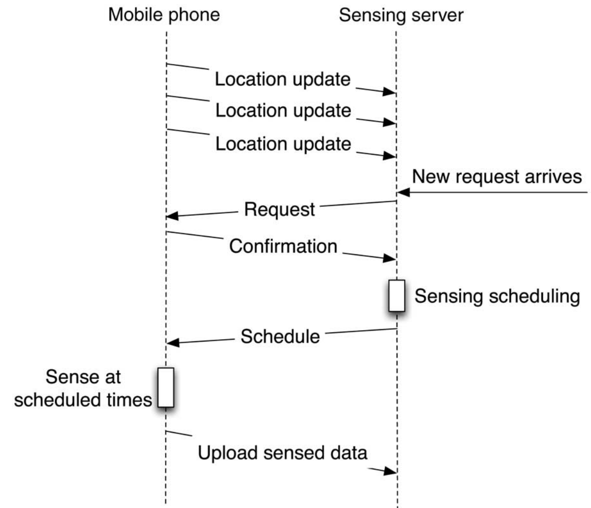


Fig. 3. Proposed sensing scheduling protocol.

Essentially, any mobility prediction algorithm can be applied here. We design a simple and effective algorithm for prediction and use it in our system. For prediction, we assume that each mobile user only moves along a roadway and only turns in an intersection. In our prediction algorithm, we use a set of evenly distributed landmarks to characterize a roadway. Currently, the distance between a pair of consecutive landmarks was set to 1 m. The historical locations reported by mobile phones (which might not be on roadways because they are not accurate) are mapped to the closest landmarks on the roadways. Then the prediction algorithm checks all the landmarks corresponding to recent reported locations and selects most recent two which happen to belong to the same roadway. In Fig. 4, the two solid circles in the center are the two selected landmarks. Using these two landmarks, the algorithm calculates the speed and direction for this mobile user. The future locations can thus be predicted by assuming that the mobile user will not change his/her speed and direction within this timeslot. Based on the prediction, the algorithm will then generate the future moving trajectory (as shown by stars in the figure), which stops if a predefined number (which was set to 10 in our simulation and field test), or an intersection

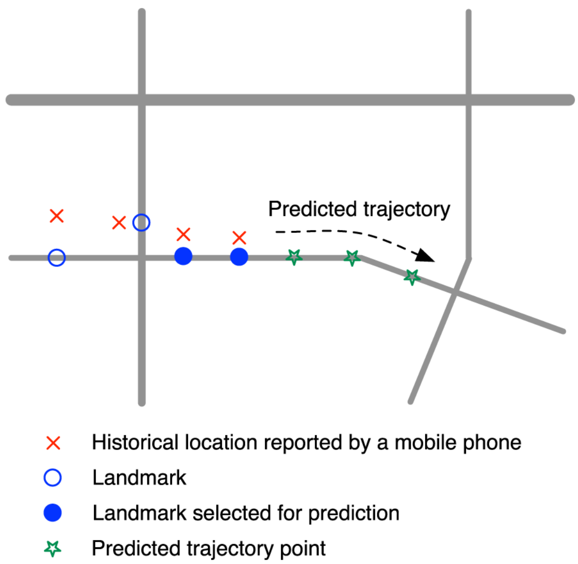


Fig. 4. Mobility prediction algorithm.

is reached. The algorithm makes no attempt to predict the trajectory after an intersection, since it cannot know to which direction the user will head.

III. Implementation

We developed an Android-based mobile crowd sensing system and implemented the proposed protocol.

In our implementation, scripts (instead of binary codes [7]) are used to describe every sensing task, which can be pushed to mobile phones for execution. We chose Lua [15] as the scripting language in our implementation. A mobile app was developed for the Android platform to execute scripts on-the-fly with the help of an interpreter and operate sensors on a mobile phone to perform requested sensing activities according to a given sensing schedule.

Our sensing server includes the following software components: 1) sensing task generator—it generates a new sensing task in a standard format based on request information collected from the web interface; 2) mobile phone scheduler—it schedules sensing activities of mobile phones for each sensing task according to the sensing scheduling protocol (discussed in Section II-D); 3) mobile phone maintainer—it keeps track of a list of mobile phones that are available for undertaking sensing tasks, and serves as an interface between a sensing server and mobile phones for information/data exchange; and 4) data manager—it uses a database to store and manage states of available mobile phones, sensing tasks, and sensed data.

We used Erlang [8] to build the sensing server. With the help of Erlang, we implemented the above-mentioned components as separate modules; each of them can be started, stopped, and restarted on its own. So when part of the system encounters a failure, we can easily isolate that specific part, fix, and restart it without having to restart the whole system, which enables high availability for our system. We selected a mature relational database, PostgreSQL [21], for data storage.

IV. Validation and Performance Evaluation

A. Simulation Results

In the simulation, the target area we chose is an area located in the center of New York (Manhattan), NY, USA, which spans three blocks from west to east with a total length of 1600 m and three blocks from south to north with a total length of 800 m, and includes the 5th, 6th, 7th, and 8th Avenues and the 44th, 45th, 46th, and 47th Streets. We used a mobility model similar to the well-known Manhattan model [1] to generate mobile users' moving trajectories. Specifically, each user is assumed to be at a random location in the target region at the very beginning. Then it randomly selects a direction and moves with a speed randomly selected from {2,4} meters per second towards an intersection. Then the user moves straight ahead with a probability of 50%, and turns left or right with equal probabilities (i.e., 25%). The trajectory of each user was constructed with points on critical locations (such as intersections and road heads) and evenly spaced sample points (1 m between two consecutive ones) between them. All the location data were collected from the Google Maps using its APIs. The deadline (the duration of the whole sensing period) was set to $T = 1800$ s. The sensing range was set to $R' = 20$ m. The target points were evenly distributed on the roadways in the target area with a distance of 40 m between a pair of consecutive points. Random errors were introduced to simulate the inaccuracy of reported locations.

In the simulation, we evaluated the performance of the greedy algorithm for E-MCSS (E-MCSS-Greedy), the greedy algorithm for F-MCSS (F-MCSS-Greedy), the protocol using E-MCSS-Greedy (E-MCSS-Protocol), and the protocol using F-MCSS-Greedy (F-MCSS-Protocol). We used the average coverage probability (i.e., the total coverage probability divided by the number of target points) as the performance metric and changed the number of mobile users from 20 to 70, with a step size of 5. In the protocol, the whole sensing period was divided into multiple timeslots with the same duration, which was set to 200 s in the simulation. $T = 1800$ s, but the first timeslot was used only for collecting user's mobility information for prediction. Therefore, there are eight timeslots for sensing. For F-MCSS-Protocol, we set each user's sensing budget in each timeslot to $b = 2$ and $b = 4$, respectively, in two scenarios. To ensure fair comparisons, for F-MCSS-Greedy, each user's total sensing budget was set to $B_i = b \times 8$, i.e., 16 and 32, respectively. Accordingly, the total sensing budget was set to $B = B_i \times K$ for E-MCSS-Greedy and the total sensing budget in each timeslot for E-MCSS-Protocol was set to $b \times K$, where K is the number of users. Moreover, we used the periodic sensing method (i.e., every mobile phone periodically senses without collaborations) as the baseline for comparison, in which each mobile phone senses every 100 s. Note that during the sensing period, some mobile users might leave the target area at certain times, which were not used for sensing any longer after their departure in the simulation.

We conducted simulation runs on 40 sets of randomly generated trajectories. The simulation results are shown in Fig. 5(a)–(c). We can make the following observations.

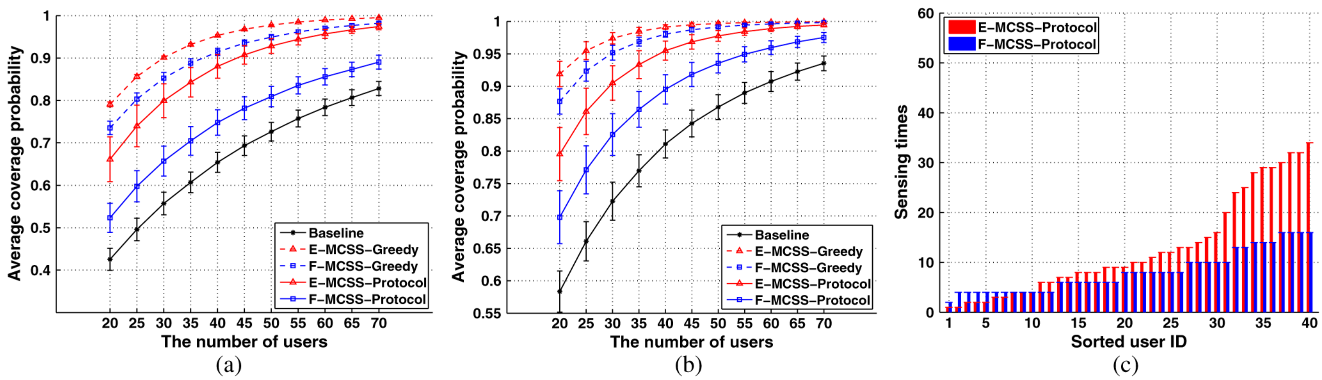


Fig. 5. Performance of the proposed algorithms and protocol. (a) Average coverage probability vs. the number of users ($b = 2$). (b) Average coverage probability vs. the number of users ($b = 4$). (c) Sensing times vs. sorted user ID ($b = 2$).

1) As expected, all the proposed approaches outperform the baseline method and E-MCSS-Greedy performs best in terms of coverage probability. On average [based on Fig. 5(a) and (b)], in terms of the coverage probability, E-MCSS-Protocol and F-MCSS-Protocol outperform the baseline method (for the case where $b = 2$) by 33.3% and 15.5%, respectively. We can also see that the performance of these two methods is comparable. In addition, we can view their performance from another perspective. In order to achieve an average coverage probability of 80%, according to Fig. 5(a), E-MCSS-Protocol needs about 30 users and F-MCSS-Protocol needs about 50 users, whereas the baseline method needs about 65 users.
2) No matter which method is used, the average coverage probability always increases with the number of users. In addition, by comparing Fig. 5(a) and (b), we can see that the higher the sensing budget (which leads to more energy consumption), the better the coverage.
3) In Fig. 5(c), we show fairness of E-MCSS-Protocol and F-MCSS-Protocol by plotting the number of sensing times of each user (the total number of users is 40) using a set of randomly generated trajectories. We can see that compared to E-MCSS-Protocol, F-MCSS-Protocol offers more even distribution of the number of sensing times among mobile users, which means better fairness. Note that the total number of sensing times given by E-MCSS-Protocol is larger than that given by F-MCSS-Protocol due to the early departure of some mobile users.

B. Experimental Results

The field test was performed on Syracuse University's campus, in which eight volunteers participated. The application is to scan WiFi signal strengths on roadways. Popular Android smartphones, including Motorola Droid, Motorola Razr, Google Nexus S, Samsung Galaxy I9000, and Galaxy S2 were used as our sensing devices. Each mobile phone used Google's location service to obtain its locations and report them to the sensing server every $T_{\rm loc} = 20$ s. *For the testing purpose*, GPS was also turned ON to collect the *actual* locations where mobile phones performed WiFi scans.

At the very beginning of our field test, each user was randomly located on a roadway in the target area. Then he/she started to walk along the roadway in a random direction at his/her regular walking speed and turned randomly in an intersection. The settings of the field experiment are summarized in Table I.

TABLE I
EXPERIMENTAL SETTINGS

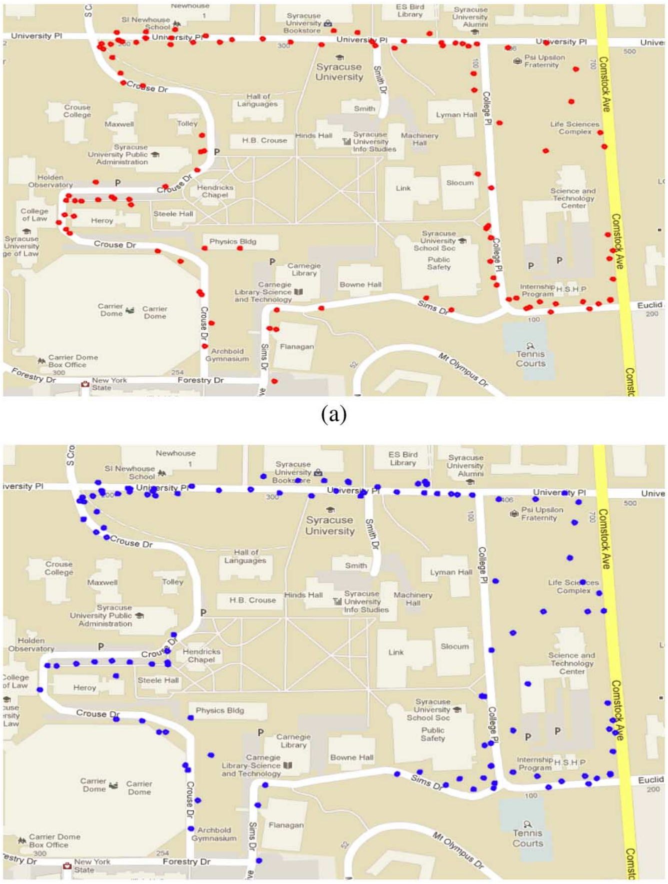
Variables	Description
$K = 8$	The number of users
$T = 1800$ s	The duration of the whole sensing period
$T' = 200$ s	The duration of a timeslot
$T_{loc} = 20$ s	Location reporting period
$b = 2$	The sensing budget for each user in a timeslot
$T_p = 100$ s	The time interval for the baseline method

We tested the proposed protocol and system in the field experiment. The sensing budgets were set in the same way as simulations. Note that in the first timeslot, each mobile phone simply reported its locations obtained from the Google's location service to the sensing server for mobility prediction without performing any WiFi scans. We plot the users' actual sensing locations (obtained from GPS) given by the proposed protocol in Fig. 6. From the figure, we can see that the locations that mobile phones sense are widely distributed over the roadways. In this way, the target points in the area are well covered, which verifies that the proposed system and protocol work well in a real environment with random mobility and unreliable wireless links.

V. Related Work

There are quite a few mobile phone/crowd sensing projects [12], focusing on different areas such as environment monitoring [18], [22], health and well being [4], social networking [17], and wireless signals [25]. Unlike these works that were focused on particular applications, we target energy-efficient mobile crowd sensing in general.

Recently, efforts have been made to develop general-purpose systems to support various mobile phone/crowd sensing applications. For example, in [7], Das *et al.* presented a platform for remote sensing using smartphones (PRISM), which allows application writers to package their applications as executable binaries and push them automatically to an appropriate set of mobile phones. The bubble sensing system proposed in [14] allows sensing tasks to be posted for specific physical locations



(a)

(b)

Fig. 6. Actual sensing locations given by the proposed protocol. (a) E-MCSS-Protocol. (b) F-MCSS-Protocol.

of interest. Cornelius *et al.* introduced AnonySense in [6], which is a sensor tasking and reporting system. Micro-blog [10] is another system for participatory sensing, where users upload blogs annotated with sensed data (e.g., photos) and their locations to a micro-blog server. In addition, Sheng *et al.* provided an overview for such systems and pointed out future research directions in [27].

However, these existing systems have the following problems: 1) PRISM [7] uses executable binaries to deliver sensing tasks to mobile phones, which is platform-dependent (Windows Mobile only) and may cause security issues and 2) energy efficiency has not been well addressed by any of them, which, however, is the main focus of this paper.

Only few recent works addressed collaborative sensing with uncontrollable mobile phones or sensors. In [29], the authors introduced mechanisms for automated mapping of urban areas that provide a virtual sensor abstraction to applications. They also proposed spatial and temporal coverage metrics for measuring the quality of acquired data. In [28], the authors proposed a protocol, Aquiba, that exploits opportunistic collaboration of pedestrians. In [26], Sheng *et al.* presented algorithms for energy-efficient sensing scheduling and showed that significant power savings can be achieved by collaborative sensing via simulations. Incentive mechanisms have been proposed in [31] to attract mobile users to participate in sensing activities. The optimization problems considered here are mathematically different from those in these related works. Specifically, energy-efficient sensing scheduling without accurate location information has not been considered by any of them.

VI. Conclusion

In this paper, we presented a probabilistic model for sensing coverage without accurate location information, based on which we formally define the E-MCSS and the F-MCSS problems. Under a strong assumption that the moving trajectories of mobile users are known in advance, we presented a $(1-1/e)$-approximation algorithm and a $1/2$-approximation algorithm to solve the E-MCSS and F-MCSS problems in polynomial time, respectively. Under realistic assumptions, we presented a GPS-less and energy-efficient protocol for sensing scheduling based on the proposed approximation algorithms. It has been shown by simulation results that the proposed protocol significantly outperforms a baseline method in terms of coverage probability and F-MCSS-Protocol (the protocol using the greedy algorithm for F-MCSS) can achieve a good tradeoff between coverage and fairness (on individual mobile phone usages). Experimental results from a field test were also presented to validate the proposed protocol.

References

[1] F. Bai, N. Sadagopan, and A. Helmy, “The IMPORTANT framework for analyzing the Impact of Mobility on Performance Of RouTing protocols for Ad hoc NeTworks,” *Ad Hoc Netw.*, Vol. 1, No. 4, pp. 383–403, 2003.

[2] *Buffer Overflow* [Online]. Available: http://en.wikipedia.org/wiki/Buffer_overflow

[3] A. Carroll and H. Gernot, “An analysis of power consumption in a smartphone,” in *Proc. USENIX Ann. Tech. Conf.*, 2010.

[4] S. Consolvo *et al.*, “Activity sensing in the wild: A field trial of ubifit garden,” in *Proc. ACM Conf. Human Factors Comput. Syst.*, 2008, pp. 1797–1806.

[5] I. Constandache, R. Choudhury, and I. Rhee, “Towards mobile phone localization without war-driving,” in *Proc. IEEE Conf. Comput. Commun. (IEEE Infocom'10)*, 2010.

[6] C. Cornelius *et al.*, “Anonysense: Privacy-aware people-centric sensing,” in *Proc. ACM Annu. Int. Conf. Mobile Comput. Netw. (ACM MobiSys'08)*, 2008, pp. 211–224.

[7] T. Das *et al.*, “PRISM: Platform for remote sensing using smartphones,” in *Proc. ACM Annu. Int. Conf. Mobile Comput. Netw. (ACM MobiSys'10)*, 2010, pp. 63–76.

[8] *Erlang Programming Language* [Online]. Available: http://www.erlang.org/

[9] *Fitbit* [Online]. Available: http://www.fitbit.com/

[10] S. Gaonkar *et al.*, “Micro-blog: Sharing and querying content through mobile phones and social participation,” in *Proc. ACM Annu. Int. Conf. Mobile Comput. Netw. (ACM MobiSys'08)*, 2008, pp. 174–186.

[11] L. Gargano and M. Hammar, “A note on submodular set cover on matroids,” *Discrete Math.*, vol. 309, no. 18, pp. 5739–5744, 2009.

[12] N. D. Lane *et al.*, “A survey of mobile phone sensing,” *IEEE Commun. Mag.*, vol. 48, no. 9, pp. 140–150, 2010.

[13] K. Lin, A. Kansal, D. Lymberopoulos, and F. Zhao, “Energy-accuracy trade-off for continuous mobile device location,” in *Proc. ACM Annu. Int. Conf. Mobile Comput. Netw. (ACM MobiSys'10)*, 2010, pp. 285–297.

[14] H. Lu, N. D. Lane, S. B. Eisenman, and A. T. Campbell, “Bubble-sensing: Binding sensing tasks to the physical world,” *J. Pervasive Mobile Comput.*, vol. 6, no. 1, pp. 58–71, 2010.

[15] *Lua Programming Language* [Online]. Available: http://www.lua.org/

[16] *MaxMind* [Online]. Available: http://www.maxmind.com/app/geolite_city_accuracy

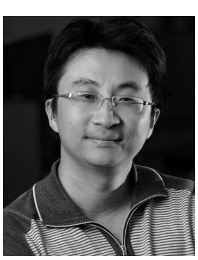
[17] E. Miluzzo *et al.*, “Sensing meets mobile social networks: The design, implementation and evaluation of the CenceMe application,” in *Proc. ACM Conf. Embedded Netw. Sens. Syst. (ACM SenSys'08)*, 2008, pp. 337–350.

[18] M. Mun *et al.*, "PEIR, the personal environmental impact report, as a platform for participatory sensing systems research," in *Proc. ACM Annu. Int. Conf. Mobile Comput. Netw. (ACM MobiSys'10)*, 2010, pp. 55–68.
[19] G. Nemhauser, L. Wolsey, and M. Fisher, "An analysis of the approximations for maximizing submodular set functions," *Math. Programm.*, vol. 14, pp. 265–294, 1978.
[20] *Andorid's Location Strategies* [Online]. Available: http://developer.android.com/guide/topics/location/strategies.html
[21] *PostgreSQL Database System* [Online]. Available: http://www.postgresql.org/
[22] R. Rana, C. Chou, S. Kanhere, N. Bulusu, and W. Hu, "Ear-phone: An end-to-end participatory urban noise mapping system," in *Proc. Int. Conf. Inf. Process. Sensor Netw. (ACM/IEEE IPSN'10)*, 2010, pp. 105–116.
[23] S. Reddy, "MobiSense: Mobile network services for coordinated participatory sensing," in *Proc. IEEE Int. Symp. Auton. Decentralized Syst. (IEEE ISADS'09)*, 2009.
[24] *Sensordrone* [Online]. Available: http://sensorcon.com/sensordrone/
[25] *Sensorly* [Online]. Available: http://www.sensorly.com
[26] X. Sheng, J. Tang, and W. Zhang, "Energy-efficient collaborative sensing with mobile phones," in *Proc. IEEE Int. Conf. Comput. Commun. (IEEE Infocom'12)*, 2012, pp. 1916–1924.
[27] X. Sheng, J. Tang, X. Xiao, and G. Xue, "Sensing as a serivce: Challenges, solutions and future directions," *IEEE Sensors. J.*, vol. 13, no. 10, pp. 3733–3741, Oct. 2013.
[28] N. Thepvilojanapong *et al.*, "Opportunistic collaboration in participatory sensing environments," in *Proc. ACM Workshop Mobility Evolving Internet Archit. (ACM MobiArch'10)*, 2010, pp. 39–44.
[29] H. Weinschrott, F. Durr, and K. Rothermel, "StreamShaper: Coordination algorithms for participatory mobile urban sensing," *IEEE Int. Conf. Mobile Ad-Hoc Sens. Syst. (IEEE MASS'10)*, 2010, pp. 195–204.
[30] D. Yang, X. Fang, and G. Xue, "ESPN: Efficientt server placement in probabilistic networks with budget constraint," in *Proc. IEEE Int. Conf. Comput. Commun. (IEEE Infocom'11)*, 2011, pp. 1269–1277.
[31] D. Yang, G. Xue, X. Fang, and J. Tang, "Crowdsourcing to smartphones: Incentive mechanism design for mobile phone sensing," in *Proc. ACM Int. Conf. Mobile Comput. Netw. (ACM MobiCom'12)*, 2012, pp. 173–184.

Xiang Sheng (S'09) received the B.E. degree in electrical engineering from the Nanjing University of Science and Technology, Nanjing, China, in 2008, the M.S. degree in electrical engineering from Syracuse University, Syracuse, NY, in 2010, and is currently working toward the Ph.D. degree in electrical engineering and computer science from Syracuse University.

His research interests include wireless networking, mobile computing, and green communications.

Mr. Sheng was the recipient of a Best Paper Award at IEEE ICC'2014.



Jian Tang (M'08–SM'13) received the Ph.D. degree in computer science from Arizona State University, Phoenix, AZ, USA, in 2006.

He is currently an Associate Professor with the Department of Electrical Engineering and Computer Science, Syracuse University, NY, USA. He has published over 70 refereed technical papers in premier journals and conferences. His research interests include cloud computing, mobile computing, big data, and green networking.

Dr. Tang received a Best Paper Award at IEEE ICC'2014 and an NSF CAREER Award in 2009.

Xuejie Xiao received the B.S. degree in software engineering from Tongji University, Shanghai, China, in 2011. He is currently working toward the Ph.D. degree at the Department of Electrical Engineering and Computer Science, Syracuse University, NY, USA.

His research interests include cloud computing, high performance computing systems, large-scale distributed systems, and big data analysis.

Guoliang Xue (M'98–SM'99–F'11) received the Ph.D. degree in computer science from the University of Minnesota, Minneapolis, MN, USA, in 1991.

He is currently a Professor of Computer Science and Engineering with Arizona State University, Phoenix, AZ, USA. He has authored or coauthored over 200 refereed technical papers, many of which appeared in top journals such as the IEEE/ACM Transactions on Networking, the IEEE Journal on Selected Areas in Communications, the IEEE Transactions on Mobile Computing, and top conferences such as ICNP, INFOCOM, MOBICOM, MOBIHOC, and NDSS. His research interests include survivability and security issues in networking, quality of service provisioning, and human centric computing with collective intelligence.

Dr. Xue was a Keynote Speaker at LCN'2011—the IEEE Conference on Local Computer Networks. He serves on the editorial boards of the IEEE/ACM Transactions on Networking and the *IEEE Network Magazine*. He served as an Associate Editor and Advisory Board Member of the IEEE Transactions on Wireless Communications. He was a TPC Cochair of IEEE INFOCOM'2010 and is a member of the INFOCOM Standing Committee. He was an IEEE Communications Society Distinguished Lecturer from January 2010 to December 2011. He was the recipient of Best Paper Awards at ICC'2012, GLOBECOM'2011, ICC'2011, and MASS'2011, as well as a Best Paper Runner-up Award at ICNP'2010.