EMC^3: Energy-Efficient Data Transfer in Mobile Crowdsensing under Full Coverage Constraint

Haoyi Xiong, Daqing Zhang, Leye Wang, and Hakima Chaouchi, *Member, IEEE*

Abstract—This paper proposes a novel mobile crowdsensing (MCS) framework called EMC^3, which intends to reduce energy consumption of individual user as well as all participants in data transfer caused by task assignment and data collection of MCS tasks, considering the user privacy issue, minimal number of task assignment requirement and sensing area coverage constraint. Specifically, EMC^3 incorporates novel pace control and decision making mechanisms for task assignment, leveraging participants' current call, historical call records as well as predicted future calls and mobility, in order to ensure the expected number of participants to return sensed results and fully cover the target area, with the objective of assigning a minimal number of tasks. Extensive evaluation with a large-scale real-world dataset shows that EMC^3 assigns much less sensing tasks compared to baseline approaches, it can save 43%-68% energy in data transfer compared to the traditional 3G-based scheme.

Index Terms—Mobile crowdsensing (MCS), energy-efficient MCS data transfer, MCS task assignment

1 Introduction

The rapid development and proliferation of sensor equipped smartphones are making Mobile Crowdsensing (MCS, also called Participatory Sensing) an effective way to enable more and more applications, ranging from environment monitoring to traffic congestion detection and social trend understanding [1]. Typically, an MCS application (or task) not only requires each participant's mobile device to possess the capability of *receiving sensing tasks, performing sensing and returning sensed results* to a central server, it also requires to receive sensed results from at least *a certain number of mobile devices covering the target area*. For example, for an air quality monitoring MCS task in a city, it needs to collect sufficient samples across the city to assess the actual air quality of the city. Apparently the success of MCS relies heavily on the active participation of mobile users, it's thus crucial to encourage mobile users to participate in MCS tasks and ensure good coverage in the crowdsensing process.

Three main factors affecting user's willingness for MCS task participation are *energy consumption*, *communication cost* and *privacy*. While the cost problem can be solved by providing users with certain incentives, the energy consumption and privacy issues should be considered and addressed in the MCS framework design. Different from the existing work in mobile crowdsensing [2], [3], [4], [5], this paper aims to design an MCS framework which can not only reduce the energy consumption of both individual and all participants in *data transfer* caused by sensing task assignment and data collection of an MCS task, but also ensure a full sensing coverage of the target region when selecting the participants, whose identities are anonymized such that one's historical call and mobility traces cannot be directly linked to the user in different MCS tasks.

The proposed framework is designed based on the following observations:

1) Energy consumption in MCS data transfer and the opportunity for energy saving. For a typical MCS application with the *task assignment*, *sensing* and *data collection* cycle, a 3G connection is needed for task assignment and data collection, respectively. Thus the energy consumption in each cycle for data transfer is equal to that consumed mainly by the two 3G connections when the size of the sensed data is small. However, if the task assignment and data collection can be piggybacked over user's daily calls or other 3G data usage, then the additional energy consumed in data transfer for an MCS task would be significantly reduced due to the reuse of established 3G connections [6], [7], [8]. This technique is termed as *Parallel Transfer*. Take the Nokia N95 phone as an example, a 3G data connection typically consumes around 12 Joules [9], while the additional energy incurred when piggybacking a data packet of 10KB over a 3G call is around 2.5 Joules (75%-90% energy saving). As a reference, sensing the noise with a microphone sensor in N95 needs about 1.114 Joules to get a valid sample of four seconds [10]. For MCS data transfer, *placing* (*making* or *receiving*) *3G calls*, *using 3G data connections*, and *initiating low-power communication alternatives* such as WiFi/Bluetooth are all opportunities to piggyback data and save energy efficiently [8], [10].

2) Delay-tolerant MCS tasks. While most existing works assume that the sensed data is sent to the server as soon as the data is produced, many MCS tasks actually don't require the sensed data to be uploaded in real-time. Quite often, certain time delay between task assignment and sensing, or between sensing and uploading is allowed. We define those MCS tasks which allow delay in sensing or uploading as *Delay-tolerant MCS applications*.

3) MCS cycles and the energy-efficient MCS data transfer mechanism. With the *delay-tolerant MCS tasks* and *parallel*

• *The authors are with CNRS UMR 5157 SAMOVAR, Institut Mines-TELECOM, TELECOM SudParis, Evry 91011, France. E-mail: {haoyi.xiong, daqing.zhang, leye.wang, hakima.chaouch}@telecom-sudparis.eu.*

Manuscript received 9 Jan. 2014; revised 8 Aug. 2014; accepted 8 Sept. 2014. Date of publication 11 Sept. 2014; date of current version 1 June 2015.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TMC.2014.2357791

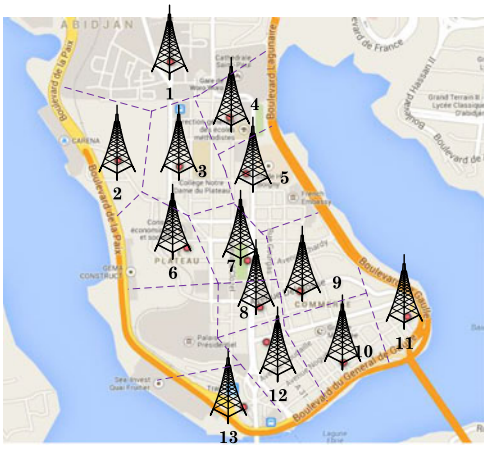


Fig. 1. Cell towers in the Abidjan CBD area.

transfer technique in mind, we could divide each MCS task into equal-length cycles, the duration of the MCS cycle depends on the actual MCS application. For instance, an environment monitoring MCS application might need to sense the air quality once per hour, thus the MCS cycle is 1 hour for this application. As in each cycle there are many users *placing* 3G calls or using 3G data connections (e.g., mobile web browsing), we could design the MCS framework such that the sensing tasks are only assigned to the call makers/receivers or 3G data users to save energy in data transfer. Likewise if the assigned users place another call or initiate a 3G data connection before the end of the cycle, they can also return the sensed results in an energy-efficient manner. This is to say, with the introduction of MCS cycles and parallel transfer technique, users with *two piggyback opportunities* such as two calls in one cycle can complete task assignment and sensed results uploading in an energy-efficient manner.

Among all possible opportunities for energy-efficient MCS data transfer, in this work we focus on the mechanism piggybacking MCS data over phone calls, with an assumption that users' call logs (including the cell tower ID and time stamp of users' each call) are accessible. Consequently, in the rest of this paper, we only study the *two-call-based MCS data transfer mechanism* leveraging two calls to complete task assignment and sensed data uploading in one sensing cycle, we further evaluate the mechanism using D4D mobile call datasets [11]. With the above preliminary in mind, to achieve energy-efficient data transfer in mobile crowdsensing, we thus assume:

- Only when a user places calls, the device could receive sensing task assignment; Only when another call comes before the end of cycle, it could return sensed results to the server (in this paper we use the *mobile device*, *mobile user* and *participant* interchangeably);
- In each cycle, one participant can receive task assignment and upload results at most once;
- In the starting cycle of each MCS task, due to user identity anonymization, there are no historical call or mobility traces for any user from other or previous MCS tasks. All users only accumulate call and mobility traces within one MCS task.

Based on the above assumptions and observations, the research objective of this work is to fulfill the following three goals in each cycle of the MCS task:

- *Ensure an expected number of participants returning the sensed results.*
- *Make sure that the returned sensed results fully cover the target sensing area.*
- *Minimize the number of total task assignments to reduce overall energy consumption.*

To further clarify the research goals, let's consider the following use case: *In the CBD area of Abidjan city in Cote d'Ivoire (around 7 km^2), there are 13 cellular towers installed in the 3G network as shown in Fig. 1. The city government, with the help of a telecom operator, launches a series of MCS tasks leveraging the 3G cellular network infrastructure. One of the MCS tasks is air quality monitoring in the CBD area, it requires to update the air quality to the citizens of Abidjan once every 2 hours (cycle) and the task lasts for two weeks. In order to provide reliable measurements, the application needs to get sensed results from at least 40 mobile users, covering all 13 cellular towers in each cycle.* Please note that, in the considered use case and the rest of this paper, we use cell towers as the coverage metrics, primarily due to two reasons: 1) The cell tower IDs of mobile phones are accessible in call logs, even though the cell tower is not the right coverage metrics for many MCS applications, the mobile phone call logs with cell tower as coverage metrics are used to illustrate the basic idea of handling coverage constraint problem in MCS applications; 2) For MCS applications such as urban air quality monitoring [12], noise level monitoring [13], etc., covering all the cell towers in a given region ensures that each part of the given area is measured with certain guarantee, even though the sampled granularity in terms of cell tower may not be the best choice. If it could be characterized more precisely, the proposed approach could be easily adapted.

With the above research goals and use case, the key issues in designing the MCS framework include:

1) Identify "candidate users" who might place two or more calls, and predict which subarea each user might cover in each MCS cycle. As only the users placing two calls can fulfill sensing task using *parallel transfer*, and some candidate users must cover the low-density call subarea, thus it's necessary to choose the right candidates based on call and mobility prediction of the current caller, to minimize redundant task assignments. Apparently, assigning sensing task to users placing one call in a cycle or to candidate users only from high-density call subareas would lead to redundant task assignments, causing big overall energy consumption.

2) Given the arrived call sequence at certain instant of an MCS cycle, estimate if the number of users assigned could expect the predefined number of returned results and cover all the subareas. As the users receiving task assignment need to wait till the next call to return sensed results, thus there is a delay between assigning tasks and receiving the expected number of results from the target area, so it's necessary to make predictions and stop unnecessary task assignments.

3) If further task assignments are still needed to achieve the goal of getting expected number of returned results and full coverage, we need to decide if the sensing task should be assigned to the current one or the future candidates. As there are more valid

candidates than needed and candidates from low-density call subareas might appear late in one cycle, we should decide the task assignment based on whether the current candidate or future candidates have higher probability of meeting the three goals.

4) Ensure the goals to be met despite the time-varying and inaccurate nature of all probability estimations. As the candidate user selection and task assignment are all based on future call and mobility predictions, while all those predictions are based on probability estimations which might not be accurate. For example, both the future call and mobility predictions are based on the historical traces, in the first MCS cycle, the prediction accuracy for both call and mobility could be very low, this will definitely cause sub-optimal decisions, leading to redundant task assignments. Fortunately, the estimation of all parameters is carried out with each incoming call, with continuous monitoring and adjustment, the system is designed to adapt itself to get both the expected number of sensed results and the full coverage, filtering out a lot of unnecessary task assignments.

In summary, the main contributions of this work are:

1) We formulate the problem of energy saving in data transfer of MCS tasks for both individual and all participants, with consideration of privacy issue as well as full coverage constraint. To the best of our knowledge, this is the first work addressing this issue. In particular, we propose to leverage the *parallel transfer* and *delay-tolerant mechanism* to achieve the energy saving purpose in MCS applications.

2) We develop a *three-step decision making process* and the related algorithms for effective task assignment in MCS applications. Specifically, we first *identify "candidate users"* who might place two or more calls and predict which subarea they might cover in each MCS cycle; Then we *judge if sufficient task assignments have been made* by considering if the number of assigned users could expect to return a predefined number of sensed results and cover all the target area; Finally, we *decide if a new sensing task should be assigned to the current one or a future candidate*, based on whether the current candidate or the future candidate has higher probability of meeting the three goals.

3) Through extensive evaluation of our proposed algorithms on the real world dataset D4D [11], which contains four-month call records of 50,000 users from Cote d'Ivoire, we verify that our proposed MCS framework EMC^3 can ensure the expected number of participants returning their sensed results with full coverage and much less redundant task assignments than baseline approaches. Through leveraging parallel transfer over 3G calls, EMC^3 reduces around 75 percent energy consumption in data transfer for a returned participant and 43-68 percent overall energy consumption in data transfer for MCS applications, such as air quality or noise monitoring at the Abidjan CBD area, compared to the traditional 3G-based scheme.

2 Related Work

There has been a lot of work about energy saving in MCS or participatory sensing, as the energy consumption required affects the battery life of mobile devices and thus user's willingness to participate in MCS tasks [1]. In this section, we would like to discuss the related work in MCS from two perspectives: one is to review the related work on energy saving of individual device, the other is on overall energy consumption reduction under coverage constraint.

2.1 Work on Energy Conservation of Individual Device

For individual mobile device, the energy consumed by an MCS task falls into the following three parts: *sensing*, *computing* and *data transfer*. In order to reduce energy consumption in *sensing*, existing approaches include adopting low power sensors [5], adapting sensing frequency [14], and inferring data rather than sensing data directly [15]. To save energy in *computing*, the MCS framework can use low power processors [16], energy-efficient sensed data processing algorithm [17] or code offloading [18]. To reduce energy consumption in *data transfer*, researchers have proposed to use low power communication methods [19], [20] and transmit less data e.g., compressing sensed results [21] or sending part of the data while deducing the rest [4], [22]. Our proposed framework EMC^3 adopts the *parallel transfer* technique [6] to reduce energy consumption in data transfer, which is complementary to the above-mentioned approaches, thus most of them can be incorporated into EMC^3 to make the mobile crowdsensing more energy-efficient.

2.2 Work on Overall Energy Saving under Coverage Constraint

For saving overall energy in an MCS task, the basic idea is to reduce energy consumption in each mobile device and deploy as less mobile devices as possible. In order to formulate the overall energy conservation as an optimization problem in mobile crowdsensing, researchers often use sensing coverage as the constraint, then the research goal becomes finding the minimal number of participants in order to cover the whole target area or a certain percentage of the target area. In [23], Reddy et al. first discuss the research challenge of balancing overall energy consumption and coverage in participatory sensing and further propose a coverage-based recruitment strategy to select the minimal number of users for reducing overall energy. The authors of [4], [24] introduce the notion of virtual sensors which intend to collaboratively infer sensing values to reduce physical and redundant sensing, they propose spatial and temporal coverage metrics for minimizing the overall energy consumption and balancing the energy consumption and the data quality. In [22], Musolesi et al. present several techniques to optimize the information uploading process for continuous sensing, they also consider the coverage and overall energy consumption in MCS.

More recently, Sheng et al. [3] propose a mechanism to reduce the overall energy consumption in mobile crowdsensing by optimizing the schedule of each sensing device, collaboratively all the mobile devices could fully cover the target region with minimal sensing energy. Slightly different from the above works, Ahmed et al. [25] propose to consider partial rather than full coverage as a sensing constraint, they have thus proposed probabilistic models to find the minimal set of participants to balance the overall energy consumption and the cover ratio of target area.

Hachem et al. [2] also use the mobility prediction to estimate the possible coverage of a certain mobile device and select a minimal number of mobile users which can cover a certain percentage of the target area, so as to minimize the overall energy consumption for sensing and service registration.

Although EMC^3 shares the same objective of *reducing the energy consumption under the coverage constraint* with the above related works, our work is different from them in the following aspects: 1) Unlike all abovementioned works which focus on reducing overall energy consumption *in sensing*, we mainly focus on reducing the overall energy consumption *in data transfer*. However, the techniques in reducing sensing energy consumption can be also applied in our framework. 2) The optimization problems considered in those related works are *mathematically different* from the problems considered here, due to different objectives and assumptions. For example, we have a different coverage definition from the related works, and we request to return a fixed number of returned results with minimal number of task assignments under full coverage. 3) All the related works intend to reduce redundancy *caused by assigning multiple sensors to acquire the same data at the same location*, assuming that each assigned participant would return sensed results; While EMC^3 attempts to minimize the *redundant task assignments* with the assumption that some assigned participants won't return sensed results back, therefore the solutions are quite different. While our work predicts both *call and mobility* to achieve full coverage, only *mobility prediction* is needed in all the related works. 4) While the evaluations of the most related works use either small scale real-world data or simulated data set, we use a *large-scale real-world* mobile phone dataset D4D to verify the effectiveness of our proposed algorithms.

3 Problem Statement

With the observations, assumptions and research goals elaborated in the introduction, the essence of the research problem of this work is to determine if a task assignment should be made, given an incoming call and historical call and location traces of a specific MCS task, in order to obtain a predefined number of returned sensed results with minimum number of task assignments under the full coverage constraint and given assumptions. While the number of task assignments and returned results are easy to count, we need to define what the full coverage of target area means.

In this work, we say that a cell tower is covered by a user when she places a call receiving a task assignment or returning sensed results to the server in the cell tower. If a user places one call in one cell tower for receiving a task assignment and another call in another cell tower for returning the sensed results, then these two cell towers are said to be covered by the user. Hence, the full coverage means that all the cell towers in the target area should be covered by at least one call for receiving task assignment or returning sensed results. Please note that the cell towers traversed by the user between the two calls are not counted in this work. In the rest of this paper, we name the participant who covers a cell tower as the *covering participant* for the cell tower. With all the above definitions, we formally formulate the MCS task assignment problem in EMC^3 as follows:

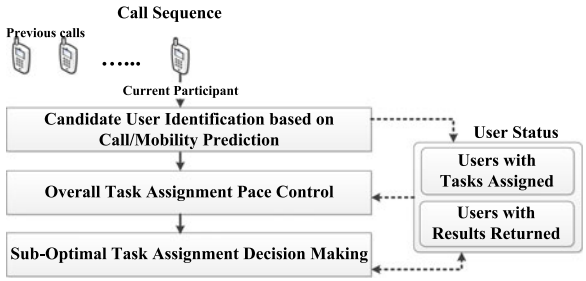


Fig. 2. The EMC^3 framework.

Given an MCS task with starting time t_0, sensing cycle duration T, the expected number of collected sensing data N_e from each sensing cycle, and a cover region pre-defined by a set of cell towers T_{WR}; Given the incoming call and all previous call traces (including the time stamped calls and cell towers associated) in the MCS task, the elapsed time t in current cycle k, we denote A_k as the set of participants who have been assigned with sensing tasks since the start of cycle k, R_k as the set of participants who have returned sensed results, and $cover_k$ as the set of cell towers that have been covered, where apparently R_k is a subset of A_k and $cover_k$ is a subset of T_{WR}. Our problem is to decide if an MCS sensing task should be assigned to the current caller, with the objective to

$$\text{minimize } |A_k|, \text{ subject to } |R_k| \geq N_e \textbf{ and } cover_k \equiv T_{WR}$$

by the end of cycle k. It is worth noting that we do not know when and where a participant would place a phone call in advance but there are sufficient number of calls covering the cell towers of the target area.

4 EMC^3 Framework and Core Algorithms

EMC^3 follows a centralized MCS system approach where a central server continuously monitors all the participants' calling activities in the target region and decides if a user should receive sensing task assignment for each incoming call. As shown in Fig. 2, EMC^3 consists of three main components, i.e., *candidate user identification*, *task assignment pace control*, and *sub-optimal task assignment decision making*; These three functional modules correspond to the *three-step working process* of EMC^3, respectively. In addition to the three functional components, EMC^3 takes the previous call traces (including the current cycle and previous cycles) as input, it also keeps the user list with task assignments as well as the user list with sensed results returned for *task assignment pace control* and *sub-optimal task assignment decision making*. In the following, we will briefly describe each of the three functional components:

Candidate user identification based on call/mobility prediction. Given an incoming call, the *candidate user identification* module first updates the call records for the user. Based on the user's historical time-stamped call and location records, the module can predict the probability of having future calls and the associated cell towers before the end of the cycle. If the user has a high probability of placing another call in the desired cell towers, and she hasn't received any task assignment in the current cycle, then she is considered as a

candidate user for further task assignment (go to next step for task assignment pace control). Otherwise, EMC^3 either collects sensed data from her (in case she received task assignment but hasn't returned results in the same cycle) or ignores her to take care of the next call.

Overall task assignment pace control. This module controls if further task assignment is still needed to fulfill the goal of getting expected number of sensed results from all the cell towers. For this purpose, the module first counts the number of *returned users* and their *covered cell towers*, collects the user list who have got task assignment but haven't returned sensed results (defined as *potential returners*), and computes their probability of returning the *missing number* of sensed results in the desired cell towers. If the number of *returned users* reaches the pre-defined value and the *returned users* fully cover all the cell towers, then the task assignment process of the current MCS cycle stops; If previous task assignments can expect to return the pre-defined number of results covering all cell towers, then no immediate assignment is needed in order to avoid redundant task assignment. If previous task assignments cannot ensure the return of expected number of results or the full coverage, then further task assignment is still needed (goes to next step for task assignment decision making).

Sub-optimal task assignment decision making. Given the incoming call and previous call records, if the task assignment pace control module informs that further task assignment is still needed, then this module decides if the current caller/receiver should be assigned with a sensing task in order to meet the three research goals. In order to make an `optimal` (`sub-optimal`) decision, this module counts the number of *returned users* and the *covered cell towers*, collects the *potential returner* list, and predicts the *future frequent callers* who haven't placed phone calls but would have higher probability of making at least two calls than the current caller making another call (defined as *future-surer candidates*). With the returned user list, *potential returner* list and *future-surer candidate* list, the module estimates if the last two sets of users can expect to return the *missing number* of predefined sensed results in the *desired cell towers*. If the probability is very high, then the task assignment is skipped for current caller and left to future users; If the last two user lists cannot ensure to get the *missing number* of sensed results in the required cell towers, the sensing task is assigned to the current caller, indicating that the current user is among the most potentially frequent callers.

In the following, we introduce the core algorithms used in the three components in detail.

4.1 Call/Mobility Prediction

We predict the call/mobility of a user based on the periodicity of previous calls and locations in historical call traces. Suppose an MCS task splits one day into M sensing cycles. Given a sensing cycle k and the elapsed time t, we model a user U_i's call/mobility pattern in cycle k by using U_i's phone call traces (including time-stamps and cell tower ids) in *corresponding cycles* of previous days. For example, to predict the call/mobility pattern of a user in current sensing cycle from 08:00 to 10:00, we will use all her previous call records during the same period 08:00-10:00. Note that the calls placed by U_i in a current cycle are also included for her call/mobility prediction.

4.1.1 Modeling Call Patterns

Assume the call sequence follows an inhomogeneous Poisson process [26], then the probability of a user U_i to place n phone calls from instant t to the end of cycle k can be modeled as:

$$P_{k,t}\{x_i = n\} = \left(\lambda_{i,k,t}\frac{\Delta t}{T}\right)^n * e^{-\lambda_{i,k,t}\frac{\Delta t}{T}}/n!, \quad (1)$$

where $\Delta t = (t_0 + K * T) - t$ denotes the *remaining time* from instant t to the end of the cycle, T is the sensing cycle duration, and $\lambda_{i,k,t}$ refers to the Poisson intensity, which is estimated as the average number of phone calls that a user U_i has placed in previous corresponding cycles, specifically it is modeled as:

$$\lambda_{i,k,t} = \frac{\text{Number of calls of } U_i \text{ in perviois corresponding cycles}}{\lceil k/M \rceil}.$$

4.1.2 Modeling Mobility Patterns

Given previous call records at sensing cycle k, a participant U_i, a set of cell towers T_{WR} and a cell tower $c_j \in T_{WR}$, we define U_i's future presence probability in cell tower c_j as the the ratio between the number of U_i's historical calls at corresponding cycles in cell tower c_j and the total number of calls at corresponding cycles, i.e.,:

$$D_k(i,j) = \frac{\text{Number of calls of } U_i \text{ in the corresponding cycles in } c_j}{\text{Number of calls of } U_i \text{ in the corresponding cycles}}.$$

If the given participant U_i hasn't placed any call in the corresponding cycles, then $D_k(i,j) = 0, \forall c_j \in T_{WR}$.

Algorithm 1. Pace Control Mechanism

```
1 if |R_k| < N_e OR cover_k ≠ T_WR then
2   computing P_fulfill
3   computing P_cover_l, for ∀c_l ∈ (T_WR − cover_k)
4   if P_fulfill < P_G1 OR
      ∃c_l ∈ (T_WR − cover_k), P_cover_l < P_G2 then
5     Goto Next Step for Further Task Assignment;
6   else
7     No Need for Further Task Assignment;
8 else STOP;
```

4.2 Overall Task Assignment Pace Control

Given the list of *potential returners* ($A_k - R_k$), the *missing number* of sensed results ($N_e - |R_k|$) and the instant time (t) in cycle k, we estimate

- $P_{fulfill}$. the probability of having at least ($N_e - |R_k|$) *potential returners* placing another call before the end of cycle k.

Given the list of *potential returners* ($A_k - R_k$), a *desired cell tower* $c_l \in (T_{WR} - cover_k)$ and the instant time (t), we estimate

- P_{cover_l}. The probability of having at least one *potential returner* placing another call to cover the cell tower c_l before the end of cycle k.

With $P_{fulfill}$ and P_{cover_l} defined, EMC^3 controls the pace of task assignment using the pseudo code in Algorithm 1, where P_{G1} and P_{G2} are two given thresholds. In this way, the key is to calculate $P_{fulfill}$ and P_{cover_l}.

4.2.1 Estimating $P_{fulfill}$

First, we define $P\{X_{k,t,1}(A_k - R_k) = N\}$ as the probability of having N out of $|A_k - R_k|$ *potential returners* placing at least another call before the end of cycle k, where $N \leq |A_k - R_k|$ (see Eq. (2)). In this way, $P_{fulfill}$ is estimated as the sum of $P\{X_{k,t,1}(A_k - R_k) = N\}$, where N is an integer ranging from the *missing number* of sensed result ($N_e - |R_k|$) to the total number of *potential returners* ($|A_k - R_k|$) (see Eq. (3)).

$$P\{X_{k,t,1}(A_k - R_k) = N\} = \sum_{\forall s \subset A_k - R_k}^{|s|=N} \prod_{\forall U_m \in s} P_{k,t}\{x_m \geq 1\} \prod_{\forall U_m \in A_k - R_k - s} (1 - P_{k,t}\{x_m \geq 1\}) \tag{2}$$

$$P_{fulfill} = \begin{cases} 0, & |A_k| < N_e \\ \sum_{N \geq N_e - |R_k|}^{N \leq |A_k - R_k|} P\{X_{k,t,1} & (A_k - R_k) = N\}, \\ & |A_k| \geq N_e. \end{cases} \tag{3}$$

Please note that, when the number of participants already assigned is less than the expected number of sensed results—i.e., $|A_k| < N_e$, then it is not possible to collect the pre-defined number of sensed results, thus $P_{fulfill} = 0$. Considering the complexity of $P_{fulfill}$ estimation, we propose an algorithm to reduce the computation complexity and time as shown in Appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TMC.2014.2357791.

4.2.2 Estimating P_{cover_l}

First, we define $COV_{k,t}(m, l)$ as the probability of a given *potential returner* U_m ($U_m \in A_k - R_k$) covering a given *uncovered cell tower* c_l ($c_l \in T_{WR} - cover_k$) before the end of cycle k. Assume U_m received the task assignment in cell tower c_{assign} ($c_{assign} \in T_{WR}$), apparently there are two possible cases: one is $c_{assign} = c_l$, the other is $c_{assign} \neq c_l$. In the case of $c_{assign} = c_l$, $COV_{k,t}(m, l)$ is equal to the probability of U_m placing at least another call before the end of cycle k (in arbitrary cell tower T_{WR}). In the case of $c_l \neq c_{assign}$, $COV_{k,t}(m, l)$ is equal to the probability of U_m placing another call in cell tower c_l before the end of cycle k. Hence we have:

$$COV_{k,t}(m, l) = \begin{cases} P_{k,t}\{x_m \geq 1\} & , \quad c_l = c_{assign}, \\ P_{k,t}\{x_m \geq 1\} * D_k(m, l), & c_l \neq c_{assign}, \end{cases} \tag{4}$$

where $P_{k,t}\{x_m \geq 1\}$ denotes the probability of U_m placing at least another call before the end of cycle k, and $D_k(m, l)$ is the probability of U_m appearing in cell tower c_l. With the above definition of $COV_{k,t}(m, l)$, P_{cover_l} can be calculated using Eq. (5) below [25]:

$$P_{cover_l} = 1 - \prod_{\forall U_m \in A_k - R_k} (1 - COV_{k,t}(m, l)). \tag{5}$$

4.3 Sub-Optimal Task Assignment Decision Making

Given the incoming call and previous call records, the key algorithms of this step include 1) identifying all *future-surer candidates*, 2) estimating if the *missing number* of sensed results can be returned from *future-surer candidates* and *potential returners*, 3) estimating if all *desired cell towers* can be covered by *future-surer candidates* and *potential returners*, and 4) sub-optimal task assignment decision making.

4.3.1 Identifying Future-Surer Candidates

Given the current caller U_i, we consider U_m as a *future-surer candidate* if:

- U_m has placed calls in previous *corresponding cycles* but has not placed any call in current cycle, **and**
- U_m has a higher probability of placing at least two calls than U_i placing at least another call, i.e., $P_{k,t}\{x_m \geq 2\} > P_{k,t}\{x_i \geq 1\}$, **or** U_m has placed more calls in any desired cell tower ($T_{WR} - cover_k$) than U_i.

Putting all the *future-surer candidates* together with regard to U_i, they are denoted as FS_{U_i}.

4.3.2 Estimating if the Missing Number of Sensed Results Can Be Returned from Future-Surer Candidates and Potential Returners

Given the set of *future-surer candidates* FS_{U_i}, the set of *potential returners* ($A_k - R_k$), and the *missing number* of sensed results ($N_e - |R_k|$), we estimate $P^*_{fulfill}$ as the probability of having at least the *missing number* of sensed results ($N_e - |R_k|$) returned from the *potential returners* and *future-surer candidates* ($(A_k - R_k) \cup FS_{U_i}$) before the end of cycle k. Apparently the estimation of $P^*_{fulfill}$ depends on the probability of each U_m returning the sensed results ($U_m \in (A_k - R_k) \cup FS_{U_i}$) before the end of cycle k, each U_m's returning probability can be computed using Eq. (6):

$$P'_{k,t}(U_m) = \begin{cases} P_{k,t}\{x_m \geq 1\}, & U_m \in (A_k - R_k) \\ P_{k,t}\{x_m \geq 2\}, & U_m \in FS_{U_i}. \end{cases} \tag{6}$$

In the case of $U_m \in (A_k - R_k)$ (belonging to the *potential returner set*), $P'_{k,t}(U_m)$ is modeled as the probability of U_m placing at least another call before the end of cycle k. In the case of $U_m \in FS_{U_i}$ (belonging to the *future-surer candidate set*), then $P'_{k,t}(U_m)$ is modeled as the probability of U_m placing at least two calls before the end of cycle k. Given each user U_m's returning probability $P'_{k,t}(U_m)$, similar to the estimation of $P_{fulfill}$ in Eq. (3), $P^*_{fulfill}$ can be computed using Eqs. (7) and (8):

$$P^*_{fulfill} = \begin{cases} 0, & |A_k \cup FS_{U_i}| < N_e \\ \sum_{N \geq N_e - |R_k|}^{N \leq |(A_k - R_k) \cup FS_{U_i}|} P\{X_{k,t,2} & (FS_{U_i}) + X_{k,t,1} \\ & (A_k - R_k) = N\}, \\ & |A_k \cup FS_{U_i}| \geq N_e. \end{cases} \tag{7}$$

$$P\{X_{k,t,2}(FS_{U_i}) + X_{k,t,1}(A_k - R_k) = N\} = \sum_{\forall s \subset (A_k - R_k) \cup FS_{U_i}}^{|s|=N} \prod_{\forall U_m \in s} P'_{k,t}(U_m) \times \prod_{\forall U_m \notin s} (1 - P'_{k,t}(U_m)). \quad (8)$$

Considering the complexity of $P^*_{fulfill}$ estimation, we use the same algorithm as shown in Appendix, available in the online supplemental material, to reduce the computation time.

4.3.3 Estimating if All Desired Cell Towers Can Be Covered by Future-Surer Candidates and Potential Returners

Given a *desired cell tower* c_l ($c_l \in (T_{WR} - cover_k)$), the set of U_i's *future-surer candidates* (FS_{U_i}), and the set of *potential returners* ($A_k - R_k$), we define $P^*_{cover_l}$: the probability of cell tower c_l to be covered by at least one participant from the set of *potential returners* and *future-surer candidates* ($(A_k - R_k) \cup FS_{U_i}$). Apparently the estimation of $P^*_{cover_l}$ depends on the probability of each U_m ($U_m \in (A_k - R_k) \cup FS_{U_i}$) covering the given cell tower c_l before the end of cycle k, the probability of each U_m's covering c_l can be computed using Eq. (9):

$$COV^*_{k,t}(m,l) = \begin{cases} COV_{k,t}(m,l), & U_m \in (A_k - R_k) \\ P_{k,t}\{x_m \geq 2\} * & \left(1 - (1 - D_k(m,l))^2\right), \\ & U_m \in FS_{U_i}. \end{cases} \quad (9)$$

In the case of $U_m \in (A_k - R_k)$ (belonging to the *potential returner set*), $COV^*_{k,t}(m,l)$ is the same as $COV_{k,t}(m,l)$. In the case of $U_m \in FS_{U_i}$ (belonging to the *future-surer candidate set*), then $COV^*_{k,t}(m,l)$ is modeled as the probability of U_m placing at least two calls (at least one of the first two calls placed in cell tower c_l), before the end of cycle k . Given the probability of each user U_m covering cell tower c_l – i.e., $COV^*_{k,t}(m,l)$, similar to the estimation of P_{cover_l} in Eq. (5), $P^*_{cover_l}$ can be computed using Eq. (10):

$$P^*_{cover_l} = 1 - \prod_{U_m \in (A_k - R_k) \cup FS_{U_i}} (1 - COV^*_{k,t}(m,l)). \quad (10)$$

4.3.4 Sub-Optimal Task Assignment Decision Making

With $P^*_{fulfill}$, $P^*_{cover_l}$ computed and two thresholds P_{G1}, P_{G2} given, EMC^3 assigns a task to the current caller (U_i) if $P^*_{fulfill}$ is lower than P_{G1}, or there exists any cell tower $c_l \in (T_{WR} - cover_k)$ having $P^*_{cover_l}$ lower than P_{G2}. The pseudo code of sub-optimal task assignment decision making is shown in Algorithm 2.

Algorithm 2. Sub-Optimal Task Assignment Decision Making Mechanism

```
1  computing P*_fulfill
2  computing P*_cover_l, for ∀c_l ∈ (T_WR − cover_k)
3  if P*_fulfill < P_G1 OR
     ∃c_l ∈ (T_WR − cover_k), P*_cover_l < P_G2 then
4    Assign the sensing task to U_i;
5  else
6    Not Assign;
```

5 Evaluation

In this section, we will report the evaluation results using the large-scale real-world call traces to verify the effectiveness for our proposed method in reducing energy consumption in data transfer for MCS tasks. We first introduce two baseline methods and the parameter settings for evaluating EMC^3 briefly. Second, we present two D4D phone call traces and the basic experiment settings. Then, the detailed evaluation results of EMC^3 with respect to the two baseline methods are presented and compared. Finally, based on the known methods in energy consumption estimation, the EMC^3 and other relevant schemes are compared in terms of energy consumption for MCS data transfer.

5.1 Baseline Methods and Parameter Settings

In our evaluation, we provide two baseline methods with respect to EMC^3:

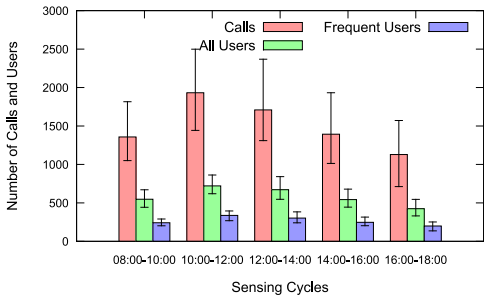
1) *Greedy*. Assigning the sensing task to each new calling user, till the expected number of sensed results are returned and all the cell towers are covered.
2) *Pace Control based Method (Pace)*. Leveraging our proposed *task assignment pace control* mechanism. If the pace control mechanism decides that further task assignment is still needed and the current caller is new in this cycle, it assigns the sensing task to the current caller.

Apparently, Greedy method is the baseline which can show the upper bound for the maximum number of task assignment and returned results, it can also provide ground truth for coverage. Compared to the Greedy method, the Pace method can show the effectiveness of pace control mechanism in reducing the redundant task assignment. The comparison between EMC^3 and Pace method shows the effectiveness of *sub-optimal task assignment decision making* mechanism in determining if the current or future callers are better candidates for task assignment, in order to avoid redundancy in task assignments. In all the experiments, we set the thresholds $P_{G1} = 99.99\%$ and $P_{G2} = (99.99\%)^{1/|T_{WR}|}$ for evaluating EMC^3 as well as Pace control based method.

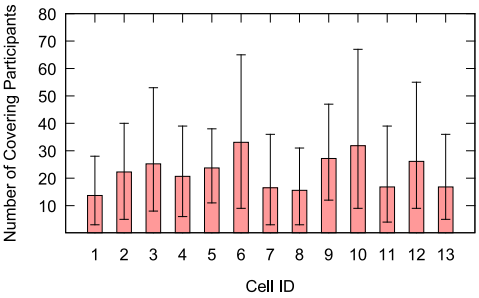
5.2 Dataset and Experiment Setups

The dataset we use in this research is the D4D dataset, which contains 50,000 users' phone call traces (each call records includes user id, call time, and cell tower) in four months from Cote d'Ivoire (where 2,000 cell towers are installed). Specifically, the 50,000 users are re-selected randomly from all the mobile users every two weeks with anonymized user ids. Thus in this study, we assume that each MCS task lasts for two weeks accordingly. Further more, we split the four-month data traces into eight two-week slots, with each two-week slot corresponding to one MCS task. And every MCS task executes five cycles every working day from 8:00 to 18:00, with each cycle lasting for two hours (i.e. 8:00-10:00,. . ., 16:00-18:00). We extract the phone call records from the CBD area (named "Plateau") and a high-end residential district (named "Cocody") of Abidjan city, and use these two call traces for evaluation:

CBD Traces—As shown in Fig. 1, the *first* target region for the MCS task execution is assumed to contain 13 cell



(a) Calls, Users and Frequent Users



(b) Covering Participants

Fig. 3. Statistics of CBD Call Traces.

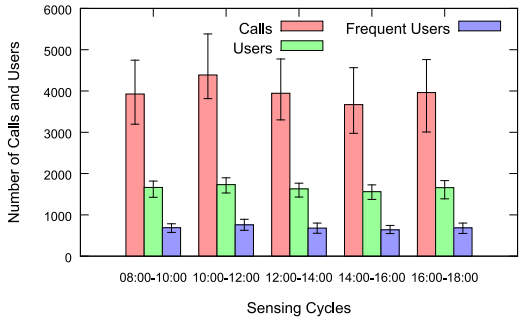
towers in the CBD of Abidjan city. For each MCS task (two-week period), about 2,000-3,000 users[1] have been found placing calls in the target region. These users are considered as the crowdsensing participants. In order to have the ground truth about the CBD region in D4D dataset, we show the number of calls, calling users, as well as the *frequent users* (placing at least two calls in a cycle) in each sensing cycle in Fig. 3a. Because the minimum number of frequent users in these cycles is 101, we thus set the expected number of returned participants (N_e) from 30 to 100. For coverage, we show the Max/Min/Avg[2] number of all covering participants found from the datasets in each cell tower per sensing cycle in Fig. 3b, extracting from the four-month dataset. It can be seen from Fig. 3b that each cell tower can be covered by at least three participants per cycle, which means that the full coverage constraint is supported by the ground truth of the D4D dataset.

Residential district traces—As shown in Fig. 4a, the *second* target region for the MCS task execution is assumed to contain 50 cell towers in an upmarket residential area (around 40 km^2) of Abidjan city. For each MCS task (two-week period), about 7,000-8,000 users have been found placing calls in the target region. In order to get the ground truth about the call traces, we show the number of calls, calling users, as well as the *frequent users* in each sensing cycle in Fig. 4b. Because the minimum number of frequent users in these cycles is 560, we thus set the expected number of returned participants $N_e = 250$ and $N_e = 500$ respectively. For coverage, we show the Max/Min/Avg number of all covering participants found from the traces in each cell

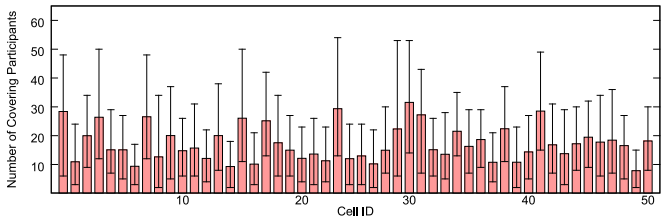
1. As a reference, there are about 7.2 million inhabitants in Abidjan, where around 75 percent inhabitants are mobile phone users [27].

2. In this paper, we name "Max/Min/Avg" as "Maximum/Minimum/Average" in short.

(a) Cell Tower Distribution



(b) Calls, Users and Frequent Users



(c) Covering Participants

Fig. 4. Statistics of Abidjan residential district call traces (best viewed in digital form).

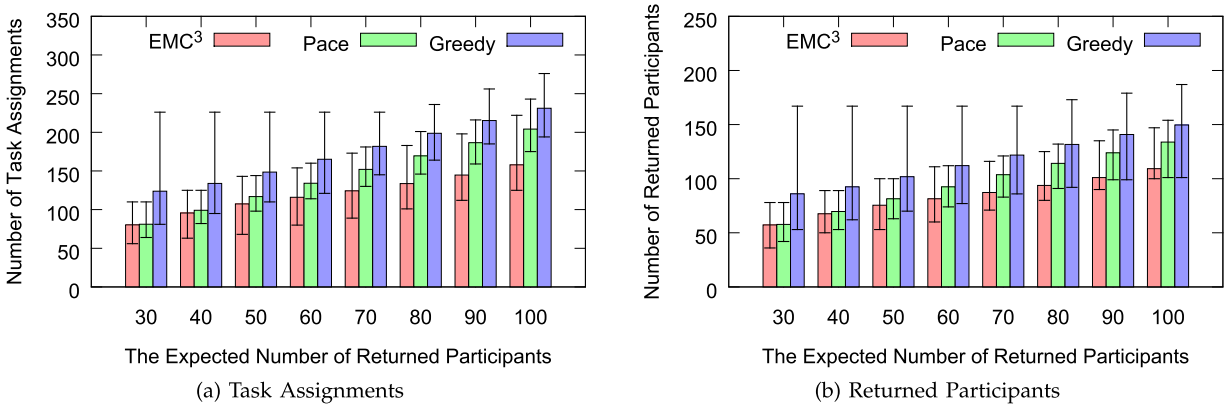
tower per sensing cycle in Fig. 4c. It can been seen that each cell tower can be covered by at least four participants per cycle, which means the call traces of Residential District can also meet the full coverage constraint. Obviously, the Residential District Traces contain more call records from more people in a larger area.

5.3 Performance Evaluation

In this part, we first compare the performance of EMC^3, Pace and Greedy methods in terms of number of task assignments, number of returned results, and coverage; Then we use an example to explain why the proposed EMC^3 outperforms Pace and Greedy method.

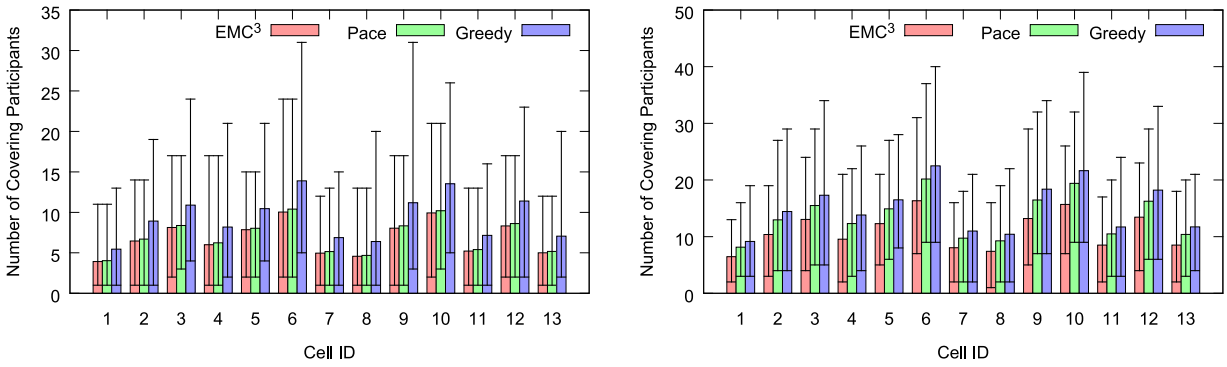
5.3.1 Performance Comparison Based on CBD Traces

In Fig. 5, we present the Max/Min/Avg number of task assignments and returned participants for the three methods under the same MCS setting, when the expected number of returned results N_e is set to vary from 30 to 100 based on CBD Traces. In order to show the coverage of the three methods, we show the Max/Min/Avg number of covering participants in each cell tower under the same MCS setting with $N_e = 40$ and 100, respectively in Fig. 6. Due to the space limit, we only select the evaluation results with $N_e = 40$ and 100. From the evaluation results shown in Figs. 5 and 6, we observe that:



(a) Task Assignments (b) Returned Participants

Fig. 5. Number of task assignments and returned participants (CBD traces).



(a) $N_e = 40$ (b) $N_e = 100$

Fig. 6. Number of Covering Participants (CBD Traces).

Task assignments. Fig. 5a shows clearly that EMC^3 assigns fewer tasks to participants than Pace and Greedy. On average, EMC^3 reduces 1-23 percent task assignments compared to Pace, and it also reduces 27-35 percent task assignments compared to Greedy method.

Returned participants. Fig. 5b shows, even in the worst case, all EMC^3, Pace and Greedy are able to collect sensed results from more than N_e participants. However, in all the cases the number of returned results is bigger than the expected number N_e, even though the number of returned results for EMC^3 is 1-18 percent fewer than Pace and 26-33 percent fewer than Greedy on average. For the Greedy and Pace methods, it's easy to understand that the big number of returned results are due to the highly redundant task assignments. For EMC^3, the reason for the big number of task assignment and returned results is mainly due to the inaccurate call/mobility prediction with limited number of call traces.

Coverage. Fig. 6 shows that any of the 13 cell towers can be covered by at least one participant with these three methods. Specifically, some cell towers have more participants than the others in a cycle. Interestingly, the distribution of covering participants in different cell towers remains more or less the same when N_e varies, and it's similar to the natural distribution of covering participants shown in Fig. 3b.

5.3.2 Performance Comparison Based on Residential District Traces

In Table 1, we present the performance comparison between EMC^3 and baselines using Residential District Traces. We count the average/minimum/maximum number of task assignments and returned participants. It is obvious that EMC^3 outperforms Pace and Greedy—EMC^3 reduces 8-18 percent task assignments compared to Pace and reduces 24-36 percent task assignments compared to Greedy; and the number of returned participants by EMC^3 is 5-15 percent and 17-33 percent less than Pace and Greedy respectively. Furthermore, in terms of coverage, all 50 cell towers are fully covered by these three methods in every sensing cycle with all N_e settings (please see also Fig. 7, where the Max/Min/Avg number of covering participants in each cell tower under the setting of $N_e = 250$ is

TABLE 1
Performance Comparison Based on Residential District Traces: EMC^3 versus Pace versus Greedy

Schemes	Task Assignments			Returned Participants		
	Avg.	Min.	Max.	Avg.	Min.	Max.
			$N_e = 250$			
EMC^3	446.6	310	979	297.2	250	574
Pace	541.5	387	1,015	352.2	262	609
Greedy	703.5	578	1,122	445.5	369	714
			$N_e = 500$			
EMC^3	821.4	695	994	507.9	500	574
Pace	887.8	756	1,120	535.6	502	714
Greedy	1075.2	967	1,194	615.5	536	718

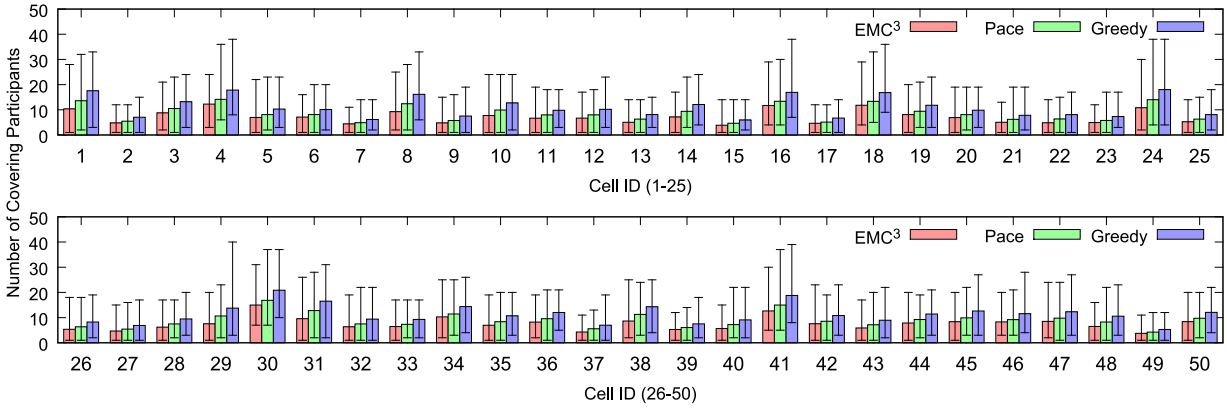


Fig. 7. Number of Covering Participants (Residential District Traces, $N_e = 250$).

shown); and the observation about the covering participants distribution is quite similar to that obtained from our experiment based on CBD call traces. From the above evaluation results, we can see that EMC^3 performs consistently better than the two baseline approaches in terms of task assignment while all the methods can achieve the goal of full coverage and collecting the predefined number of sensed results, when the target area and number of participants are different.

5.3.3 Case Study and Analysis

In order to gain more insights about the observed phenomena, we would like to show the actual task assignment process using the three methods and the Residential District call traces in sensing cycle 16:00-18:00, on 14 December 2011, where N_e is set to 250. Fig. 8 shows the actual task assignment traces of EMC^3, Pace and Greedy in the top part of the diagram (with the total number of task assignments $|A_k|$ listed), the number of covered cell towers in the middle ($|cover_k|$), and the number of returned results in the bottom (R_k). From Fig. 8, we can observe the detailed differences among EMC^3, Pace and Greedy methods, including:

Pace versus Greedy. In the beginning of the cycle, both Pace and Greedy methods assign sensing tasks to each new caller/receiver, but Pace stops assigning tasks at 16:22 when only 112 participants return their sensed results and 46 cell towers are covered. Obviously, Pace method doesn't make any further task assignments if the assigned participants are estimated to meet the requirements of covering all cell towers and collecting an expected number of sensed results. Greedy, however, stops making new task assignment at 16:39 when a total of 250 participants return their sensed results and all 50 cell towers are covered. The Pace method stops 17 minutes earlier than the Greedy method, which causes 233 less redundant task assignments and 141 less unnecessary returned results. Apparently, it's all due to the

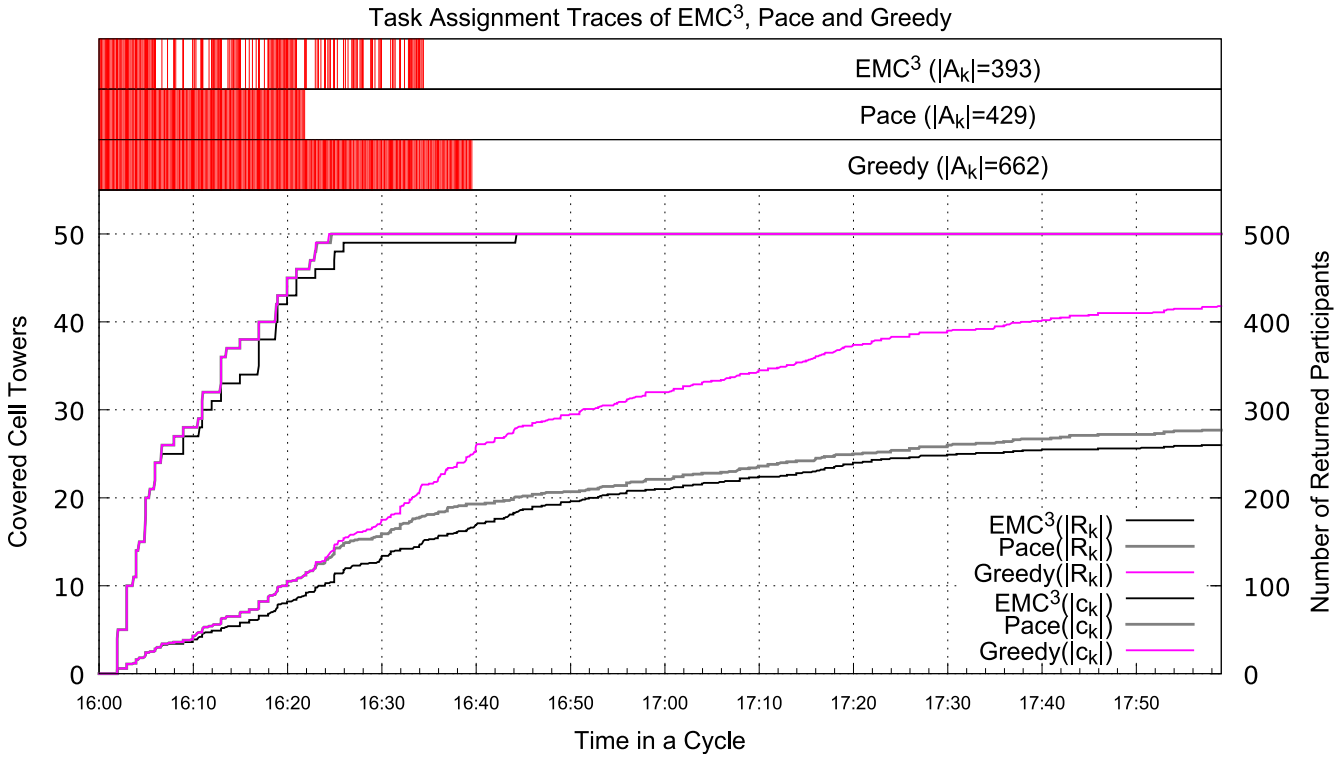


Fig. 8. Task assignment process in the case study.

TABLE 2
Energy Consumption Computation Models

Schemes	Energy Consumption
3G-based scheme	$N_e * (12 + 12) = 24 * N_e$
Parallel+3G-based scheme	$N_e * (3 + 12) = 15 * N_e$
EMC^3, Pace and Greedy	$\|A_k\| * 3 + \|R_k\| * 3$

pace control mechanism based on future call/mobility prediction.

EMC^3 versus Pace. In the beginning of the cycle, also EMC^3 assigns sensing tasks to each new caller/receiver like Pace, because the number of task assignments is much lower than the expected number of returned results N_e (250). But with the number of task assignments and returned results increasing, EMC^3 begins to select only "*frequent callers/receivers*" who have high probability of covering "*desired cell towers*" for task assignment, while Pace continues to assign sensing tasks to each new caller/receiver until the number of task assignments made is estimated to ensure receiving the expected number of sensed results covering all cell towers. In the case of Fig. 8, Pace stops assigning tasks at 16:22 when 112 participants return their sensed results and 46 cell towers are covered. EMC^3, however, stops making new task assignment at 16:34 when a total of 152 participants return their sensed results and 49 cell towers are covered. The EMC^3 method stops 12 minutes later but assigns 36 less redundant tasks than Pace. Apparently, EMC^3 outperforms Pace because of its *decision making* mechanism for task assignment, which is based on the prediction of call/mobility for both participants with task assigned and future callers/receivers.

5.4 Energy Conservation Comparison

After getting the number of task assignments and returned results with EMC^3, we would like to estimate how much energy our proposed EMC^3 scheme can save in data transfer, comparing to the following 3G-based MCS schemes:

1) *3G-based Scheme*. Receives the task assignment by establishing a new 3G connection, and returns the sensed results by establishing another 3G connection.
2) *Parallel+3G-based Scheme*. Receives a task assignment when the participant places a phone call through parallel data transfer, and returns the sensed results by establishing a new 3G connection.

Because no redundant task assignment is needed to collect sensed results with the above two schemes, we thus assume that only N_e participants from the 13 cell towers are selected to perform the MCS sensing task. Based on the literature [6] about the mobile phone energy consumption estimation method, we model the overall energy consumption in data transfer for MCS tasks by using the formulas listed in Table 2. Here the energy consumption estimation is based on the setting of Nokia N95 and the simple assumption that the data packets for task assignment or sensed results are small (less than 10 KB each). Considering the MCS applications such as air quality monitoring and environment noise monitoring, this assumption is reasonable and the energy estimated using the formula could serve as a reference for comparison purpose.

TABLE 3
Energy Consumption Comparison: 3G-Based versus Parallel +3G-based (P+3G) versus EMC^3 versus Pace versus Greedy

N_e	3G (J)	P+3G (J)	EMC^3 (J)	Pace (J)	Greedy(J)
CBD Traces					
30	720	450	*412.55*	416.47	**629.77**
40	960	600	*489.64*	506.36	**679.33**
50	1,200	750	*548.57*	594.95	**751.11**
60	1,440	900	*592.20*	679.57	831.69
70	1,680	1,050	*634.71*	766.97	910.92
80	1,920	1,200	*682.34*	850.90	991.24
90	2,160	1,350	*737.41*	931.62	1068.25
100	2,400	1,500	*801.48*	1013.90	1142.43
Residential District Traces					
250	6,000	3,750	*2231.4*	2681.1	3447
500	12,000	7,500	*3897.9*	4270.2	5072.1

Table 3 shows each scheme's average energy consumption per sensing cycle with varied N_e settings for both CBD and Residential District Traces. As can be seen from Table 3, EMC^3 can save 43-68 percent energy on average, compared to the 3G-based scheme; It can save 8-48 percent energy compared to the Parallel+3G-based scheme. Till now we are assuming that the number of expected sensed results is small, if the MCS application needs to recruit hundreds of participants and collect sensed data for many cycles a day, then the total energy saving would be significant. Interestingly, if we compare the EMC^3, Pace, Greedy with the Parallel+3G-based scheme, we can see that EMC^3 outperforms all the other schemes in all the conditions, but the Greedy method consumes more energy than the Parallel+3G-based scheme when $N_e < 50$ (using CBD traces). In summary, all the above evaluation and analytical results show the effectiveness of EMC^3 in reducing the energy consumption in data transfer for both individual and all MCS participants.

5.5 Real-Time Performance Analysis

As the decision for task assignment should be made immediately when a participant places/receives a phone call, in this section we would like to investigate if the proposed EMC^3 algorithm can be executed in the real-time setting. Thus, we first compute EMC^3's *response time*—i.e., the duration from the initial of a call (from/to a participant in the target region) to the time when the decision of task assignment is made; and then, based on the computed *response time*, we estimate EMC^3 *maximum throughput* [28]—i.e., the maximum number of mobile users allowed in the MCS system. We carry out experiments using a common laptop with an Intel Core i7-2630QM Quart-Core CPU and 8G memory. EMC^3 algorithm is implemented with the Java SE platform and is running on a Java HotSpot(TM) 64-Bit Server VM; and the implementation details are given in Appendix, available in the online supplemental material.

In order to compute the *response time* and *maximum throughput* in the realistic deployment condition, we build an EMC^3 simulator consisting of two phases:

TABLE 4
EMC^3 Average Response Time and the Estimated Maximum Throughput

	Response Time (10^{-3} sec.)		Max. Throughput (calls/sec.)	
N_e	filter	process	filter	process
CBD Traces				
30	0.0076	1.7795	131578.95	561.96
40	0.0078	1.9925	128205.13	501.88
50	0.0079	2.6816	126582.28	372.91
60	0.0080	3.1746	125000.00	315.00
70	0.0080	3.9928	125000.00	250.45
80	0.0080	5.0457	125000.00	198.19
90	0.0080	5.9189	125000.00	168.95
100	0.0080	6.9771	125000.00	143.33
Residential District Traces				
250	0.0413	268.2682	24213.08	3.73
500	0.0417	475.6196	23980.82	2.10

1) *Filter*—When a mobile user makes/receives a phone call, the system checks if the call is made/received in the target region and if the user is in the list of MCS participants; and all these operations are implemented as a simple DB query based on an embedded database. In this phase, EMC^3 identifies participants in the target region from all calls; and if the calling user is not a participant or the call is not made/received in the target region, then EMC^3 filters out the call immediately.
2) *Process*—Given a participant making/receiving a phone call, this phase executes the EMC^3 three-step task assignment decision making algorithm to decide if the participant should receive a task assignment.

Table 4 presents the average *response time* and the estimated *maximum throughput* in both phases based on different call traces and MCS task settings. Even when EMC^3 is used to monitor the Residential District, it only spends averagely no more than 0.0417 milliseconds[3] in the "*filter*" phase, which means EMC^3 is able to handle $23,980$ calls every second. As a reference, according to the D4D dataset,[4] we estimate there are approximately 1,800 calls made/received by all mobile phone users from the whole Cote d'Ivoire every second. Furthermore, EMC^3 requires averagely 0.475 seconds to complete the three-step task assignment decision making process using the Residential District traces where $N_e = 500$, which means EMC^3 is able to make decision for 2.1 incoming calls from MCS participants every second under the given condition, where *Pace* on average requires 0.076 seconds to complete the task assignment decision making process and is able to handle 13 incoming calls per second under the same setting. As a reference, even in the busiest time slot (i.e., 10:00-12:00 in working days) of Residential District, there are 0.77 calls averagely made/received by MCS participants every second. All above estimation shows that, with a high-performance server EMC^3 can easily support an larger target region than either CBD area or Residential District in real-time; and the response time of EMC^3 can be controlled to a certain value if each server is in charge of a fixed geographical area.

3. The time consumed in communication and networking has not been taken into account here; because actually EMC^3 is assumed to be deployed on telecom operator's network.

4. D4D dataset contains call traces of 0.3 percent randomly-sampled nationwide mobile phone population; and the average number of calls per second in D4D dataset is 5.4 calls/sec. Thus, we estimate the the average number of calls per second as $5.4/(0.3\%) = 1{,}800$ calls/sec.

6 Discussion

In this section, we discuss issues which are not reported or addressed in this work due to space and time constraint, these could be added to our future work.

Cold start problem. In the first day of an MCS task, as there are no historical call records due to the privacy consideration, the prediction for *frequent callers* and *future surer candidates* won't be accurate. Thus EMC^3 has the "cold start" problem which makes it perform the same as Pace method in sensing cycles of the first day, gradually with the accumulation of historical call records, EMC^3 performs better and better. The detailed evaluation results are not reported here due to space limit, but will be reported in future work.

Prediction and parameter adaption. As the performance of EMC^3 depends on the prediction accuracy of call/mobility prediction and the parameter setting used in the algorithm, in this study we currently use a simple prediction algorithm and a fixed set of parameter setting in all the sensing cycles, in future work we plan to study adaptive task assignment pace control and decision making strategies, and design advanced call/mobility prediction methods.

Sensing coverage. Due to the limitation of the D4D dataset, we can only measure one's coverage at the cell tower level; and the cell towers traversed by users between two calls are not accessible in this work. Apparently, if the user's mobility traces can be obtained continuously at fine granularity, we might consider the coverage of users more precisely.

Aggregating multiple energy-efficient strategies. In addition to piggybacking 3G data transfer over 3G calls or data packets, other data transfer methods, e.g., transferring data via WiFi/Bluetooth, also consumes less energy in data transfer, compared to common 3G-based solutions. Besides, there exist a wide range of techniques, such as adopting low-power consumption sensors or energy-efficient sensing techniques, that can save energy in the MCS tasks. In our future work, we intend to study an integrated MCS framework aggregating multiple energy-saving strategies to minimize the energy consumption in a holistic manner.

Enabling ultra-large scale crowdsensing. The evaluation result shows that EMC^3 is able to handle a large area—with tens of cell towers installed and thousands of participants making/receiving phone calls—in the real-time, while securing the data collection from hundreds of participants and under the full coverage constraint. When nation scale crowdsensing is needed, we can just divide the whole nation into multiple sub-areas and deploy multiple EMC^3 servers to manage each sub-area collaboratively. Apparently, in this way, EMC^3 can scale easily without performance issues.

7 Conclusion

In this paper, we have investigated the problem of reducing energy consumption of both individual user and all participants in data transfer caused by task assignment and data collection of MCS tasks, considering the user privacy issue, minimal number of task assignment requirement and sensing area coverage constraint. This problem is motivated by the needs of encouraging more mobile users to participate in urban-scale crowdsensing applications. To address the problem, we propose a novel MCS framework called EMC^3, leveraging a proposed delay-tolerant MCS setting, the parallel transfer technique, and a three-step process for task assignment. Evaluations with a large-scale real-world dataset show that our proposed EMC^3 framework outperforms the baseline approaches, and it can reduce 43-68 percent overall energy consumption in data transfer compared to the 3G-based solution.

References

[1] R. K. Ganti, F. Ye, and H. Lei, "Mobile crowdsensing: Current state and future challenges," *IEEE Commun. Mag.*, vol. 49, no. 11, pp. 32–39, Nov. 2011.

[2] S. Hachem, A Pathak, and V. Issarny, "Probabilistic registration for large-scale mobile participatory sensing," in *Proc. PerCom*, 2013, vol. 18, pp. 22.

[3] X. Sheng, J. Tang, and W. Zhang, "Energy-efficient collaborative sensing with mobile phones," in *Proc. INFOCOM*, 2012, pp. 1916–1924.

[4] D. Philipp, J. Stachowiak, P. Alt, F. Dürr, and K. Rothermel, "Drops: Model-driven optimization for public sensing systems," in *Proc. PerCom*, 2013, vol. 18, p. 22.

[5] G. Cohn, S. Gupta, T. J. Lee, D. Morris, J. R. Smith, M. S. Reynolds, D. S. Tan, and S. N. Patel, "An ultra-low-power human body motion sensor using static electric field sensing," in *Proc. Ubicomp*, 2012, pp. 99–102.

[6] J. K. Nurminen, "Parallel connections and their effect on the battery consumption of a mobile phone," in *Proc. Consumer Commun. Netw. Conf.*, Jan. 2010, pp. 1–5.

[7] H. Xiong, L. Wang, and D. Zhang, "Eemc: An energy-efficient mobile crowdsensing mechanism by reusing call/sms connections," in *Proc. NetMob*, 2013, pp. 323–329.

[8] N. D Lane, Y. Chon, L. Zhou, Y. Zhang, F. Li, D. Kim, G. Ding, F. Zhao, and H. Cha, "Piggyback crowdsensing (pcs): Energy efficient crowdsourcing of mobile sensor data by exploiting smartphone app opportunities," in *Proc. 11th ACM Conf. Embedded Netw. Sen. Syst.*, 2013, p. 7.

[9] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "Energy consumption in mobile phones: A measurement study and implications for network applications," in *Proc. 9th ACM SIGCOMM Conf. Internet Measurement Conf.*, 2009, pp. 280–293.

[10] Y. Wang, J. Lin, M. Annavaram, Q. A. Jacobson, J. Hong, B. Krishnamachari, and N. Sadeh, "A framework of energy efficient mobile sensing for automatic user state recognition," in *Proc. MobiSys*, 2009, pp. 179–192.

[11] V. D. Blondel, M. Esch, C. Chan, F. Clerot, P. Deville, E. Huens, F. Morlot, Z. Smoreda, and C. Ziemlicki, "Data for development: The d4d challenge on mobile phone data," CoRR abs/1210.0137, 2012.

[12] Y. Zheng, F. Liu, and H.-P. Hsieh, "U-air: When urban air quality inference meets big data," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2013, pp. 1436–1444.

[13] T. Liu, Y. Zheng, L. Liu, Y. Liu, and Y. Zhu, "Methods for sensing urban noises," Tec. Rep. MSR-TR-2014-66, May 2014.

[14] N. Roy, A. Misra, C. Julien, S.K. Das, and J. Biswas, "An energy-efficient quality adaptive framework for multi-modal sensor context recognition," in *Proc. PerCom*, 2011, pp. 63–73.

[15] D. Gordon, J. Czerny, T. Miyaki, and M. Beigl, "Energy-efficient activity recognition using prediction," in *Proc. Int. Semantic Web Conf.*, 2012, pp. 29–36.

[16] B. Priyantha, D. Lymberopoulos, and J. Liu, "Littlerock: Enabling energy-efficient continuous sensing on mobile phones," *IEEE Pervasive Comput.*, vol. 10, no. 2, pp. 12–15, Apr.–Jun. 2011.

[17] D. Chu, N. D. Lane, T. T Lai, C. Pang, X. Meng, Q. Guo, F. Li, and F. Zhao, "Balancing energy, latency and accuracy for mobile sensor data classification," in *Proc. SenSys*, New York, NY, USA, 2011, pp. 54–67.

[18] K. K Rachuri, C. Efstratiou, I. Leontiadis, C. Mascolo, and P. J. Rentfrow, "Metis: Exploring mobile phone sensing offloading for efficiently supporting social sensing applications," in *Proc. PerCom*, 2013, vol. 18, p. 22.

[19] F. Ferrari, M. Zimmerling, L. Mottola, and L. Thiele, "Low-power wireless bus," in *Proc. SenSys*, 2012, pp. 1–14.

[20] D. Puccinelli and S. Giordano, "Connectivity and energy usage in low-power wireless: An experimental study," in *Proc. PerCom Workshops*, 2013, pp. 590–595.

[21] D. Akimura, Y. Kawahara, and T. Asami, "Compressed sensing method for human activity sensing using mobile phone accelerometers," in *Proc. Int. Conf. Netw. Sensing Syst.*, 2012, pp. 1–4.

[22] M. Musolesi, M. Piraccini, K. Fodor, A. Corradi, and A. T Campbell, "Supporting energy-efficient uploading strategies for continuous sensing applications on mobile phones," in *Proc. Pervasive*, Jan. 2010, pp. 355–372.

[23] S. Reddy, D. Estrin, and M. Srivastava, "Recruitment framework for participatory sensing data collections," in *Proc. Pervasive*, 2010, pp. 138–155.

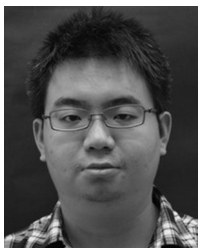
[24] H. Weinschrott, F. Durr, and K. Rothermel, "Streamshaper: Coordination algorithms for participatory mobile urban sensing," in *Proc. MASS*, 2010, pp. 195–204.

[25] A. Ahmed, K. Yasumoto, Y. Yamauchi, and M. Ito, "Distance and time based node selection for probabilistic coverage in people-centric sensing," in *Proc. SECON*, 2011, pp. 134–142.

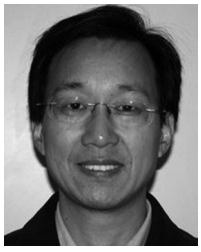
[26] J. Weinberg, L. D. Brown, and J. R. Stroud, "Bayesian forecasting of an inhomogeneous poisson process with applications to call center data," *J. Amer. Statist. Assoc.*, vol. 102, no. 480, pp. 1185–1198, 2007.

[27] Infoasaid, Telecommunications overview of cote d'ivore, 2013.

[28] "Leonard Kleinrock", *Queueing Systems: Volume 2: Computer Applications*, vol. 82. New York, NY, USA: Wiley, 1976.



Haoyi Xiong received the BEng degree in electrical engineering from Huazhong University of Science and Technology, Wuhan, China, and the MSc degree in information technology from Hong Kong University of Science and Technology, Hong Kong. He is currently working toward the PhD degree at the Institut Mines-Télécom, TELECOM SudParis and the Université Pierre et Marie Curie, Paris, France. His research interests include mobile crowdsensing and ubiquitous computing. He received the Best Paper Award from the IEEE Ubiquitous Intelligence and Computing Conference in 2012, and is currently a student member of the IEEE Computer Society and Communications Society.



Daqing Zhang received the PhD degree from the University of Rome "La Sapienza", Rome, Italy, and the University of L'Aquila, L'Aquila, Italy, in 1996. He is a professor at the Institut Mines-Télécom, TELECOM SudParis, Paris, France. His research interests include large-scale data mining, urban computing, context-aware computing, and ambient assistive living. He has published more than 180 referred journal and conference papers, all his research has been motivated by practical applications in digital cities, mobile social networks and elderly care. He is an associate editor for four journals including the *ACM Transactions on Intelligent Systems and Technology*. He has been a frequent Invited speaker in various international events on ubiquitous computing. He is the winner of the Ten Years CoMoRea Impact Paper Award at IEEE PerCom 2013, the Best Paper Award at IEEE Ubiquitous Intelligence and Computing Conference in 2012, and the Best Paper Runner Up Award at Mobiquitous 2011.

Leye Wang received the BSc and MSc degrees in computer science from Peking University, Beijing, China. He is currently working toward the PhD degree at the Institut Mines-Télécom, TELECOM SudParis and the Université Pierre et Marie Curie, Paris, France. His research interests include mobile crowdsensing and ubiquitous computing.

Hakmia Chaouchi received the PhD degree in telecommunications and networking in 2004. She is currently a professor at the Institut Mines-Télécom, TELECOM SudParis, Paris, France. Previously, she was a research fellow at Paris VI University, and also joined the Center for Telecommunication Research of London (CTR) as a research associate. Her research work is related to wireless and mobile networks and Security. She is a member of the IEEE, active in standardization activities of the IETF and the IEEE and published her research work in specialized conferences and journals.