EEMC: Enabling Energy-Efficient Mobile Crowdsensing with Anonymous Participants

HAOYI XIONG, DAQING ZHANG, LEYE WANG, and J. PAUL GIBSON,
Institut Mines-Télécom, TELECOM SudParis, France
JIE ZHU, Intel Corporation, USA

Mobile Crowdsensing (MCS) requires users to be motivated to participate. However, concerns regarding energy consumption and privacy—among other things—may compromise their willingness to join such a crowd. Our preliminary observations and analysis of common MCS applications have shown that the data transfer in MCS applications may incur significant energy consumption due to the 3G connection setup. However, if data are transferred in parallel with a traditional phone call, then such transfer can be done almost "for free": with only an insignificant additional amount of energy required to piggy-back the data—usually incoming task assignments and outgoing sensor results—on top of the call. Here, we present an *Energy-Efficient Mobile Crowdsensing* (EEMC) framework where task assignments and sensing results are transferred in parallel with phone calls. The main objective, and the principal contribution of this article, is an MCS task assignment scheme that guarantees that a minimum number of anonymous participants return sensor results within a specified time frame, while also minimizing the waste of energy due to redundant task assignments and considering privacy concerns of participants. Evaluations with a large-scale real-world phone call dataset show that our proposed *EEMC* framework outperforms the baseline approaches, and it can reduce overall energy consumption in data transfer by 54–66% when compared to the 3G-based solution.

Categories and Subject Descriptors: C.2.4 [**Distributed System**]: Distributed Applications

General Terms: Human Factors, Algorithms, Performance

Additional Key Words and Phrases: Mobile crowdsensing, energy efficiency, anonymous participants, task assignment decision making

ACM Reference Format:
Haoyi Xiong, Daqing Zhang, Leye Wang, J. Paul Gibson, and Jie Zhu. 2015. EEMC: Enabling energy-efficient mobile crowdsensing with anonymous participants. ACM Trans. Intell. Syst. Technol. 6, 3, Article 39 (April 2015), 26 pages.
DOI: http://dx.doi.org/10.1145/2644827

1. INTRODUCTION

Mobile Crowdsensing (MCS)—a term coined by Ganti et al. [2011]—is becoming increasingly popular as the number of mobile devices equipped with sensors (including phones, tablets, media players, games, and leisure/sports electronic devices) shows dramatic growth. Facilitated by the widespread adoption of sensor-equipped smartphones, MCS has been successfully adopted to enable an ever-increasing number of sensing applications, ranging from highway congestion detection [Thiagarajan et al. 2009] to

This work is supported by the EU FP7 SOCIETIES Project under grant No. 257493.
Authors' addresses: H. Xiong, D. Zhang, L. Wang, and J. P. Gibson, CNRS SAMVORA UMR 5157, Institut Mines-Télécom, TELECOM SudParis, 9 rue Charles Fourier, 91011 Evry Cedex, France; emails: haoyi.xiong, daqing.zhang, leye.wang, j.p.gibson}@telecom-sudparis.eu; J. Zhu, Intel Corporation, 2200 Mission College Blvd Santa Clara, California, 95054 United States; email: j.z.zhu@intel.com.

social trend understanding [Rachuri et al. 2013] and urban noise pollution/air quality monitoring [Rana et al. 2010; Dutta et al. 2009]. A main area of research in this field is concerned with enabling distributed monitoring applications that do not rely on a dedicated sensor network infrastructure, but where the crowdsensing communication is facilitated by an already existing network between devices (mobile phones) that are participating in the sensing tasks [Guo et al. 2014].

Typically, an MCS task is designed to collect sensing results from a specified number of participants in a certain time duration. For example, the air quality of the central business district in Abidjan City is monitored by an MCS application, which collects 40 samples of air quality sensed by different participants in the district every 2 hours. Each of the MCS participants receives a sensing task assignment, then executes it, and finally returns the sensing results. As a consequence, the air quality result sensed by participants in the most recent 2-hour period can be used to estimate and update the aggregated air quality index.

It is clear that user participation is necessary for successful mobile crowdsensing. However, two main factors are known to compromise the users' willingness to become part of a crowd:

—*Privacy:* Due to the privacy concerns, a user may not be willing to participate in all MCS tasks and may wish to anonymize herself in each MCS task in which her device participates. To ensure privacy and, as a consequence, to encourage participation, there must be no way to link a participant to her records in previous MCS tasks.
—*Energy Consumption:* The energy consumption of MCS on mobile devices may drain the battery and as such might discourage user participation. The energy consumption of an MCS task can be viewed locally by each individual crowd member or globally from the point of view of the whole crowd. *Individual Energy Consumption* is concerned with the energy consumed by the MCS task in the battery of each individual participant's mobile device, and this depends on the way that the MCS task executes on the device. The *Overall Energy Consumption* is concerned with the total energy consumed by all crowd members.

In the research being presented, we are motivated to minimize the total energy consumption of the complete MCS task through reduction of energy consumption of individual crowd members. Furthermore, we aim to achieve this goal without sacrificing the anonymity requirement.

1.1. Proposed Research: Backgrounds, Assumptions, and Objectives

In terms of energy conservation of MCS applications on mobile device, three main components—*data transfer* [Ferrari et al. 2012; Puccinelli and Giordano 2013; Akimura et al. 2012; Musolesi et al. 2010], *sensing* [Gordon et al. 2012; Ramos et al. 2011], and *computation* [Chu et al. 2011; Ra et al. 2012b]—have been the focus of study. Different from the existing work in energy-efficient mobile crowdsensing mechanisms (or frameworks) [Hachem et al. 2013; Sheng et al. 2012; Philipp et al. 2013; Cohn et al. 2012], this article aims at designing a novel Energy-Efficient Mobile Crowdsensing (EEMC) framework that addresses three aspects of the problem in an innovative manner. The mechanism will (i) minimize overall energy consumption due to data transfer, (ii) guarantee that the required number of sensor results will be returned during each cycle, and (iii) maintain the anonymity of users who have participated at any point in the lifetime of the crowdsensing activity. Our research is based on a number of well-justified assumptions:

(1) *Connection Setup Cost, and Energy Conservation in MCS Data Transfer:* Recent studies on energy consumption in a range of different devices note that a

smartphone, operating on a 3G network, typically needs to consume "12 Joules before the first byte can be sent" [Thiagarajan et al. 2012; R. Pease 2013]. The energy consumption for small data transfer ($<$10KB) is mainly concerned with establishing (and closing) the 3G connection and is also fixed at around 12 Joules [Balasubramanian et al. 2009]. This is coherent with our previous study [Xiong et al. 2013a] on air quality sensing, where we observed that when task assignments and the results of the common MCS tasks are relatively simple and the transferred data is quite small ($\leq$10KB), then the energy consumption of data transfer to receive a task assignment and return the result is also fixed at approximately 12 Joules.

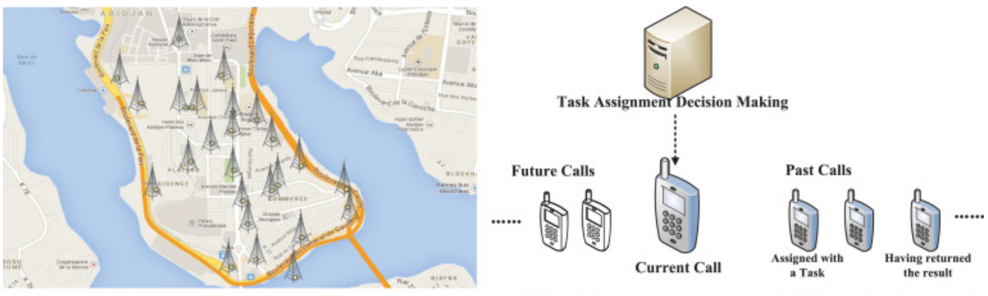
(2) *Parallel Transfer and Energy-Efficient MCS Data Transfer:*If a mobile phone receives the task assignment and uploads the sensed result in parallel with the user's regular phone calls, then the additional energy consumed in data transfer for an MCS task would be significantly reduced thanks to reuse of the already established 3G connections [Nurminen 2010; Xiong et al. 2013a]. This type of technique, which piggybacks data over connections established by voice calls or other 3G mobile applications, is known commonly as *Parallel Transfer*. Taking the Nokia N95 phone as an example, a 3G data connection typically consumes around 12 Joules (which is consistent with our first assumption), whereas the additional energy incurred when piggybacking a data packet of 10KB over a 3G call is around 2.5 Joules (which corresponds to a 75%–90% reduction in energy consumption). As an interesting comparison, sensing the noise with a microphone in the same phone requires about 1 Joule in order to get a valid sample [Wang et al. 2009].

(3) *Receive-Sense-Return Cycles and Delay-Tolerant MCS:* To support MCS applications, many different task assignment schemes [Reddy et al. 2010; Jayaraman et al. 2012; Xiao et al. 2013; Sherchan et al. 2012; Ra et al. 2012a] have been proposed. All these schemes structure mobile crowdsensing applications (on mobile devices) into three main stages "Receive—Sense—Return" (or "recruiting—sensing—uploading" in Ra et al. [2012a]). In the first stage, the mobile device receives task assignment from the central server then executes the sensing task during the second stage and returns the sensed results back to the central server in the third and final stage. A wide range of MCS tasks (a good example is the previously mentioned air quality sensing application) can be completed successfully, provided all mobile devices can go through these three stages within a specified time frame (delay) for each single task [Wang et al. 2013].

(4) *Two-Call-Based MCS Mechanism for Cyclic Sensing Tasks:* Considering the *delay-tolerant nature* of many MCS tasks, it is a reasonable assumption that we can divide an MCS task into equal-length (*Receive-Sense-Return*) cycles. In each sensing cycle, the central server attempts to collect sensing results from a required number of participants. With *parallel transfer* in mind, we can significantly reduce energy consumption in data transfer of a sensing cycle if we are able to assign sensing tasks to the mobile phone users who will place (make or receive) *two or more phone calls in the cycle*. These users receive task assignments and return their sensed results by piggybacking the data transfer on top of the calls through the *parallel transfer* approach.

In summary, to enable energy-efficient mobile crowdsensing with a *Two-Call-Based MCS Mechanism*, our initial research makes the assumptions that:

—Each MCS task lasts for a limited duration and involves a series of sensing cycles;
—All participants receive task assignments and return sensing results only when they are involved in calls;
—In each cycle, a participant will be assigned tasks no more than once;



(a) CBD Area of Abidjan (b) An Example of Sequential Task Assignment

Fig. 1. The use case of Abidjan's CBD area.

—Due to privacy concerns, all participants will be anonymized for each MCS task in such a way that we cannot link any participant to records of her previous MCS tasks.

Based on these assumptions, our research proposes an MCS task assignment mechanism that meets two objectives:

(1) *to ensure the required number of participants returning the sensing results within the cycle*, and
(2) *to minimize the number of redundant task assignments*.

To further illustrate the proposed *research assumptions and objectives*, let us reconsider the aforementioned air quality sensing use case. An environmental nongovernmental organization (NGO) in Ivory Coast, with the help of a local telco, launches an air quality monitoring MCS task in Abidjan City's CBD region where 25 cell towers are installed (see Figure 1(a)). In order to provide the timely air quality sensed results to the citizens of Abidjan city, the MCS task is designed to update the air quality reading once every 2 hours (i.e., a sensing cycle lasts for 2 hours). In order to provide reliable measures, the application is designed to secure the data collection from a minimum number (e.g., 80) of mobile users in the target area per sensing cycle. In order to facilitate the task assignment, as shown in Figure 1(b), *EEMC* is deployed on a central server that continuously monitors all mobile users' calls in the target region, analyzes the call activities of MCS participants, and decides, for each incoming call, if a participant placing (making or receiving) the call should be assigned with a sensing task. Please note that only when a participant makes/receives a phone call in the target region can she receive the task assignment or return the sensed result. In this way, tasks are assigned in a *sequential manner* as new calls are established, tasks assigned, and sensed results returned.

1.2. Research Challenges and Our Contributions

To achieve the proposed research objectives and validate them through a realistic use case, we address the following key ***technical challenges***:

—*Next-call prediction for the new arrival caller / callee based on accumulated call traces:* It is not possible to know in advance which of the crowdsensing participants will be involved in (two) phone calls during a particular sensing cycle. Thus, we need an effective method for predicting possible participation based on the participant's previous call history. However, due to the anonymization requirements, we cannot link a user with her phone call records during previous MCS tasks. Thus, there needs to be a method to predict the future phone call patterns of users using their accumulated history restricted to the current task.

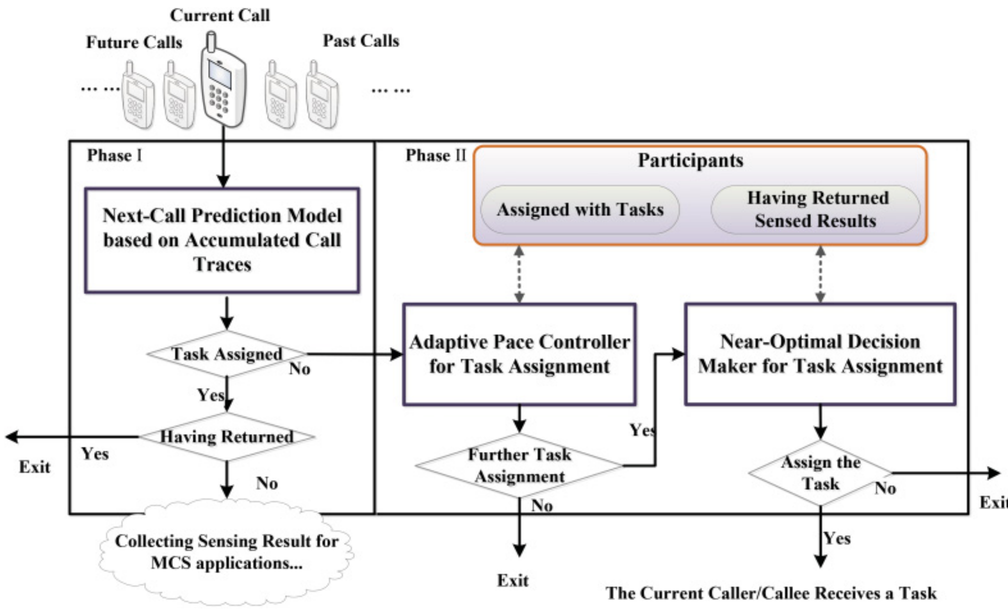

Current Call
Future Calls
Past Calls
Phase I
Next-Call Prediction Model based on Accumulated Call Traces
Task Assigned
No
Yes
Having Returned
Yes
Exit
No
Collecting Sensing Result for MCS applications...
Phase II
Participants
Assigned with Tasks
Having Returned Sensed Results
Adaptive Pace Controller for Task Assignment
Near-Optimal Decision Maker for Task Assignment
Yes
Further Task Assignment
No
Exit
Assign the Task
No
Exit
Yes
The Current Caller/Callee Receives a Task

Fig. 2. The two-phase task assignment framework.

—*Dynamically decide whether further task assignment is needed:* No method for call prediction can be perfect. As a consequence, tasks may be assigned to participants (based on their predicted call patterns) who fail to be involved in the minimum two calls required for the *"receive"* and *"return"* stages in the sensor cycle. To mitigate this problem, we propose assigning redundant tasks in such a way that the required number of results will always be returned even if an individual participant's call behavior is not as predicted. To avoid energy waste, the redundant task assignments should be *as few as possible*. The key decision that has to be made is concerned with how to update task assignments (if it all) when a new call is established during a single cycle.

—*Current calling user vs. future users; a nontrivial tradeoff:* Simple analysis would suggest that it is a good strategy to assign a task to any user who has just established a call (caller or callee), provided that she has not already been assigned a task and provided that further task assignments are needed. However, this may not be a good approach if this user has a low chance of being involved in a second call before the current cycle is complete. The decision should not be made in a `local manner`: It is better to compare the probability of the user meeting the two-call per cycle requirement with the global probability set of meeting the same requirement for all other crowd members (i.e., the participants having not placed any calls in the current cycle but with higher probabilities of placing two calls before the end of the cycle).

In this article, we propose a two-phase approach (illustrated by the process shown in Figure 2) to address the above-mentioned challenges. Consider the situation where a user is making or receiving a phone call; our first phase queries and updates her mobile phone call traces and *identifies* whether she is a *candidate for task assignment* based on *phone call prediction*. In the second phase, with a user for whom we haven't yet assigned any task in the current cycle, a *two-step* decision-making process is proposed to determine whether or not we should assign a task to her. The first step (using the *Adaptive Pace Controller for Task Assignment* component) decides *if further task* *assignments are needed based on tasks already assigned and the participants having already returned their sensing results*, and the second step (using *Near-Optimal Decision Maker for Task Assignment*) decides *if the current caller/callee should receive the task assignment through comparison with potential callers/callees in the time remaining of the current cycle*. The detailed contributions of this article are:

(1) First, motivated by saving energy in data transfer of MCS tasks for both individual participants and the whole crowd, we propose a novel mobile crowdsensing framework *EEMC* leveraging both the *parallel transfer* mechanism and the *Receive-Sense-Return cycle* pattern while also respecting the requirement for anonymity. Furthermore, we investigate and formulate the technical problem inside *EEMC*—a task assignment decision making problem—with *minimal number of task assignments* as the goal and the *predefined number of returned sensed results* as the constraint. To the best of our knowledge, this is the first work which addresses the issue of energy-efficient MCS data transfer in the proposed way.
(2) Second, we develop a two-step decision-making process and algorithms to control the task assignments. When the proposed algorithm makes a decision on task assignment, it considers four types of participants: (i) the calling user, (ii) the participants already assigned with tasks, (iii) the participants having already returned sensing results, and (iv) the future users who are (potentially) going to make two phone calls. Although this algorithm is designed for *EEMC*, other MCS frameworks with a similar optimization goal (but which do not assume that each assigned participant will return his/her sensed result) - can also benefit from application of the algorithm.
(3) Third, we evaluate *EEMC* on the Data for Development (D4D) dataset [Blondel et al. 2012] containing 4-month call detail records of Ivory Coast citizens. The result shows that *EEMC* can guarantee that the required number of participants return their sensing results while making fewer redundant task assignments than the baseline schemes. When we consider 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,the reduction is quite significant. In our case study, *EEMC* reduces energy consumption in data transfer by approximately 75% for a specific participant, with a global reduction of 54–67% for the whole crowd.

In summary, the rest of this article is organized as follows: Section 2 introduces the related work of our research; Section 3 formulates the research problem of our proposed work; Section 4 describes the *EEMC* framework and the skeleton algorithm; Sections 5, 6, and 7 present the core algorithms of *EEMC*; Section 8 introduces the baselines and dataset we used to evaluate *EEMC*; Section 9 reports on the validation of *EEMC* based on evaluation using the D4D dataset; and Section 10 discusses several open issues in our research and summarizes our conclusions.

2. RELATED WORK

In this section, we briefly review the main work related to the research being presented, focusing on four key aspects: mobile crowdsensing applications and frameworks, schemes to reduce individual energy conservation, schemes to reduce overall energy consumption, and mobile phone call prediction models.

2.1. Mobile Crowdsensing Applications and Frameworks

There has been much recent research leading to the development of many different mobile crowdsensing applications and services; for example automated recognition of human activities and context using sensor data [Lane et al. 2011], automated modeling of location characteristics [Chon et al. 2012] and linking such location semantics to user profiles [Isaacman et al. 2011], mapping network cells to geographic locations [Ficek et al. 2012], social interaction and collective behavior sensing [Rachuri. et al. 2011; Zheng and Ni 2012], mobile object discovery [Weinschrott et al. 2011] in urban areas, and road traffic/public transport monitoring [Mathur et al. 2010; Jiang et al. 2011].

To support the above-mentioned applications, many different mobile crowdsensing frameworks [Reddy et al. 2010; Jayaraman et al. 2012; Xiao et al. 2013; Sherchan et al. 2012] have been proposed. For example, Xiao et al. [2013] designs a framework to deploy MCS applications on mobile devices in order to scale the MCS system; Reddy et al. [2010] propose a framework selecting the MCS participants from volunteers before MCS task execution, where the participant selection is based on mobility data mining and reputation modeling for volunteers; Sherchan et al. [2012] introduces CAROMM, an MCS data collection framework based on mobile data mining in order to reduce the data transmission for results uploading while maintaining the accuracy of collected results; and Jayaraman et al. [2012] further develop CAROMM and provide a real-time context-aware MCS framework delivering integrated sensed results to MCS end users. Ra et al. [2012a] have presented a rapid prototyping framework called “Madusa” for mobile crowdsensing. The proposed framework structures mobile crowdsensing into three main stages: “recruiting-sensing-uploading” (which is equivalent to the three steps of “receive-sense-return” of our proposed mechanism). As a consequence, our approach can coherently support a wide range of mobile crowdsensing applications developed within this three-step framework.

Different from all previous work, which assumes that each assigned participant would return sensed results, *EEMC* assumes that assigned participants may not be able to return sensed results. This is a much more realistic assumption because it can, among other things, cope with a common scenario of a participating user’s phone being turned off in the middle of a cycle (perhaps due to the battery losing charge). In order to manage this more realistic model of the crowd of user participants, a more complex allocation algorithm based on redundancy needs to be used. However, redundancy increases energy consumption. Thus, the research challenge is to have a “fault tolerant” allocation mechanism that attempts to minimize the number of *redundant task assignments*.

2.2. Energy Saving for Individual MCS Device

As previously mentioned, the energy cost for a mobile device to perform a sensing task can be generally divided into three parts: for sensing, computation, and data transfer. As a consequence, most research on energy conservation focuses on each of these parts:

(1) To reduce the energy cost for sensing, there are many proposals ranging from the adoption of low-power sensors [Cohn et al. 2012; Larson et al. 2011] and adaptive sensor schedulers [Kjærgaard et al. 2011], to using sensing data predictors [Jiang et al. 2011; Gordon et al. 2012].

(2) To save the energy cost for computing, mobile sensing systems have turned to using low-power processors [Ra et al. 2012b] and reducing computation workloads by leveraging energy-efficient sensing data processing algorithms [Chu et al. 2011; Frank 2011] or offloading mechanisms [Ramos et al. 2011].

(3) To reduce the energy cost for data transfer, three lines of research have been conducted:

—Using low-power wireless communication [Puccinelli et al. 2012; Brown et al. 2007] can directly reduce the energy consumption of data transfer. Our research follows this approach by leveraging the parallel transfer with voice call [Nurminen 2010] as a low-power communication method.

—Using mobile nodes as relays [Puccinelli et al. 2012; Pásztor et al. 2007] to carry and forward data between sensing devices and the server can save energy since multihop relaying may still cost less than uploading data directly to the server.
—Transferring less sensing data can also save energy. The compression of sensing data [Soroush et al. 2008] can reduce the data size directly. Furthermore, strategies exist for minimizing data transfer by communicating only unpredictable data while inferring the predictable data [Musolesi et al. 2010]. These methods may consume more energy during computation, so they require a careful tradeoff to make the whole system more energy-efficient.

Finally, energy-harvesting mobile sensing systems [Smith et al. 2006] have been studied to function with battery-free platforms.

2.3. Overall Energy Saving for MCS Task

For saving overall energy in an MCS task [Reddy et al. 2010; Weinschrott et al. 2010; Philipp et al. 2013; Musolesi et al. 2010; Sheng et al. 2012; Ahmed et al. 2011; Hachem et al. 2013], the common objective is to reduce energy consumption in each mobile device and deploy the smallest number of mobile devices possible. Although EEMC shares the same objective of *reducing overall energy consumption*, our work is significantly different due to the following aspects:

(1) In order to formulate the overall energy conservation as an optimization problem in mobile crowdsensing, previous approaches use sensing coverage as the fundamental constraint: The research goal then addresses the problem of finding the minimal number of participants needed to cover a set of target areas [Reddy et al. 2010; Weinschrott et al. 2010; Philipp et al. 2013] or a certain percentage of the target areas [Ahmed et al. 2011; Hachem et al. 2013]. In contrast, *EEMC* considers the number of sensed results returned in one target area to be the fundamental constraint. The optimization problem in this case is based on a very different mathematical model, one whose goal is to return at least a predefined number of sensor results with a minimal number of task assignments. Of course, it may be possible to combine our proposed technique with existing work in order to minimize the number of selected participants with respect to a combination of both constraints. However, this is out of the scope of the work being presented in this article.
(2) Unlike approaches that focus on reducing overall energy consumption *in sensing* [Musolesi et al. 2010; Philipp et al. 2013; Sheng et al. 2012; Hachem et al. 2013], we focus on reducing the overall energy consumption *in data transfer*. However, the techniques in reducing sensing energy consumption can also be applied in our framework and would complement our approach.

Finally, the validation approaches used in previous papers use either *small-scale* real-world data or a large scale *simulated* dataset. We argue that there are *weaknesses* in both these types of evaluation approaches, and we adopt a *large-scale real-world* approach using the mobile phone dataset D4D to verify the effectiveness of our proposed algorithms.

2.4. Mobile Phone Call Prediction

Phithakkitnukoon and Dantu [2011] propose leveraging phone call prediction to support future ubiquitous computing applications and present an adaptive call predictor [Phithakkitnukoon et al. 2011] based on the historical phone call behaviors of users. We use the predictability of future phone calls of users to inform the task assignment decision process. In addition, we propose a means to guarantee the required number of participants returning sensor data even when the prediction model is not accurate.

3. PROBLEM FORMULATION

An MCS task consists of a sequence of sensing cycles—assumed to be of the same length/frequency—with each cycle requiring a predefined number of sensing data to be collected. This expected number is the most important target in data collection because sensing data processing can be compromised if insufficient updated data are available. For simplicity, we assume that the expected number of sensing data requirement is constant throughout the task and between cycles.

In this article, the MCS tasks are treated as independent of each other in order to respect the privacy protection policy. Individual calling history information of mobile users should not be shared among MCS tasks. However, during an individual MCS task, the calling history of a different group of users can be recorded, but the record will expire when the MCS task ends. In order to collect a set of sensing data from a single mobile user in one cycle, it is necessary and sufficient that the user be involved in two calls: one call for assigning a task from the server and the other for returning sensing data. Also, no mobile user in a sensing cycle can be assigned the task of collecting sensing data more than a single time. With these conditions in mind, we formally formulate the problem as follows.

Given an MCS task with starting time t_0, sensing cycle T, and the expected number of sensing data N_e from a sensing cycle, we record the timestamps and participants making/receiving phone calls from t_0. We denote A_k as the set of mobile users who have been assigned with sensing tasks since the start of cycle k and R_k as the set of mobile users who have returned sensing results, where R_k is always a subset of A_k. Every time a participant makes/receives a phone call in the sensing cycle k, our problem is to decide whether to assign a task to the participant. *The goal of task assignments* is to:

$$\text{minimize } |A_k|, \text{ subject to } |R_k| \geq N_e$$

by the end of cycle k. It should be noted that, because we cannot know in advance who is going to place another call, we cannot statically optimize the task assignment process. Therefore, the dynamic decision making for task assignments is based on a phone call history and prediction model. In this way, our research decomposes the original task assignment problem into two subproblems: *phone call prediction* and the *task assignment decision making based on the prediction*.

4. *EEMC* FRAMEWORK AND SKELETON ALGORITHM

As shown in Figure 2, *EEMC* consists of two main phases: *Candidate User Identification Based on Call Prediction* and *Two-Step Decision-Making Process for Task Assignment*. These two phases are designed to solve the *two subproblems* for task assignment decision making, respectively. In the rest of this section, we briefly describe each of the two phases.

4.1. Phase I - Candidate User Identification based on Call Prediction

Given an incoming call, *Phase I* of *EEMC* first checks if the caller is in the MCS participant list. If so, it will update the call traces of the *current caller* and identify *if the current caller is a candidate for task assignment* through predicting her future calls. Phase I has a simple design to be implemented as a single core functional module:

—**Next-Call Prediction Model Based on Accumulated Call Traces.** With historical call traces of the current caller as the input, a Predictive Model estimates the probability of the user placing another phone call in the *remaining time* (from the current time to the end of cycle).

If the *current caller* has a high probability of placing another call and has not yet received any task assignment in the current cycle, then *EEMC* deems that the user is a suitable candidate for task assignment and goes to the second step for task assignment decision making. If the *current caller* has received the sensing task assignment but hasn't returned the sensed result, then *EEMC* collects the sensed result from her. If she has already returned the sensed result or is not in the selected MCS participant list, then *EEMC* skips the call and exits the assignment process.

4.2. Phase II - Two-Step Decision-Making Process for Task Assignment

Given the *current caller* who has been identified as a candidate for task assignment (by Phase I), *Phase II* firstly decides (i) *if* EEMC*need make further task assignment(s)* and, if so, then (ii) it decides if the current caller should receive the task assignment. The Phase II design is based on two functional modules, one for each step of the decision making process:

—**Adaptive Pace Controller for Task Assignment.** Given the list of *participants already assigned* (A_k) and the list of *participants already returned* (R_k), *EEMC* estimates the probability of having a *missing number* ($N_e - |R_k|$) of potential returners ($A_k - R_k$) placing another call before the end of current sensing cycle. If the probability is higher than the given *success probability* P_s, then we decide the *tasks already assigned* are able to ensure *the expected number* of participants returning and *further task assignments are not needed immediately*. If the probability is lower than the given success probability, then *EEMC* goes to the second step for decision making of task assignment.

—**Near-Optimal Decision Maker for Task Assignment.** Given the state and history of all known participants, *EEMC* identifies the *future candidate users* who haven't placed any call in the current cycle but who are likely to place two calls before the end of the current cycle. Then, from this set of future users, *EEMC* predicts the users who have higher probability of placing two calls than the *current caller* placing another call. (We name this set the **future-surer candidates**.) With the sets of *potential returners* and *future-surer candidates* as inputs, *EEMC* estimates the probability of having a *missing number* of participants (from the two input sets) returning the sensed results. If the probability is higher than the given threshold (P_s), then there exists a sufficient number of better candidates in future and *EEMC* skips the *current caller* and leaves the sensing task to future candidates. If the probability is lower than the given threshold, then *EEMC* assigns the sensing task to the *current caller*.

With the two steps just described, *EEMC* assigns tasks to those participants who have the "*higher probabilities*" of placing another call to return their sensing results, and stops making further task assignment immediately when it predicts the tasks already assigned can secure the expected number of participants returning. Heuristically, the proposed method can minimize the total number of task assignments.

Following the above-mentioned two-phase framework, we design and implement the task assignment algorithm of *EEMC*. The skeleton of the *EEMC* algorithm is shown in Algorithm 1, where the variables are defined in Tables I and II. We will describe each module in the design of the *EEMC* algorithm in the following sections.

5. NEXT-CALL PREDICTION MODEL BASED ON ACCUMULATED CALL TRACES

EEMC predicts the call of a mobile user dependent on the periodicity of past calls in recorded call traces. Assume an MCS task parts one day into M sensing cycles. Given a sensing cycle k and the elapsed time t, we build a user U_i's call model in cycle k by mining U_i's call traces (including timestamps and cell tower ids) in *corresponding cycles* of previous days. For instance, to predict the call of a user in the current sensing cycle from 08:00 to 10:00, all her past call records throughout the same period

ALGORITHM 1: The Skeleton of EEMC Algorithm

```
Input: $M$, $k$, $A_k$, $R_k$,$U_i$,$c_j$,$t$,$S_{k,t}$, $S_1$,...$S_{k-1}$, and $P_s$
Output: {true,false}–Assign or Not
begin
    /* Phase I: Candidate User Identification based on Call Prediction */
    update_Call_Model($U_i, t, k$); // Predictive Model based on Accumulated Call Traces
    if $U_i \in A_k$ then
        if $U_i \notin R_k$ then
            collect_Sensing_Result($U_i, R_k$);
        end
        return false;
    end
    /* Phase II: Two-step Decision Making Process for Task Assignment */
    if $|R_k| < N_e$ then
        // Step 1: Pace Controller for Task Assignment
        $\mathrm{P}_{fullfill} \leftarrow \mathrm{prob}_{fulfill}(A_k, R_k, N_e, t)$;
        if $P_{fullfill} < P_s$ then
            // Step 2: Near-Optimal Decision Maker for Task Assignment
            if $k \le M$ then
                // Cold-start
                if $P_{k,t}\{x_i \ge 1\} > P_{k,t}\{x_i \ge 0\}$ then
                    $A_k \cup \{U_i\} \rightarrow A_k$;
                    return true;
                else
                    return false;
                end
            else
                // future-surer user based selection
                $FS_{U_i} \leftarrow$ futureSurer($U_i, S_1..S_{k-1}, S_{k,t}$);
                $\mathrm{P}^*_{fullfill} \leftarrow \mathrm{prob}^*_{fulfill}(A_k, R_k, FS_{U_i}, N_e, t)$;
                if $P^*_{full} < P_s$ then
                    $A_k \cup \{U_i\} \rightarrow A_k$;
                    return true;
                else
                    return false;
                end
            end
        end
    end
end
```

08:00–10:00 will be adopted. Note that the calls made/received by U_i in the current cycle are likewise incorporated for her call prediction.

5.1. Probabilsitic Model of Phone Calls

Assuming the call sequence follows an inhomogeneous Poisson process [Weinberg et al. 2007; Lin 1997], then the probability of a user U_i placing n phone calls from instant t to the end of cycle k can be modeled as:

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

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.

Table I. Symbols and Definitions

Symbols	Definitions
t_0	The starting time of an MCS task;
T	The duration of a sensing cycle;
N_e	The expected number of returned participants
k	The index of a specific cycle;
t	The elapsed time during cycle k, where $t \in [t_0 + (K-1)T, t_0 + K * T)$;
A_k	The set of participants already assigned with tasks in the cycle k;
R_k	The set of participants having already returned sensing results k;

Table II. Variables Used in *EEMC* Algorithms

Symbols	Definitions
$S_{k,t}$	The set of participants who make/receive phone calls from the start of cycle k to t, where $t \in [t_0 + (k-1) * T, t_0 + k * T)$;
S_k	The set of participants who make/receive phone calls throughout the whole cycle k;
$C_{i,k,t}$	The number of calls made/received by user U_i from the start of cycle k to t, where $t \in [t_0 + (k-1) * T, t_0 + k * T)$;
$C_{i,k}$	The number of calls made/received by user U_i throughout the whole cycle k;
M	The MCS task consists of M cycles in a day;
$P_{k,t}\{x_i = n\}$	The probability of U_i making/receiving n calls from time t to the end of cycle k, where $t \in [t_0 + (k-1) * T, t_0 + k * T)$;
FS_{U_i}	The set of future-surer users of U_i, where U_i makes/receives a phone call at t of cycle k, $\forall U_j \in FS_{U_i}, P_{k,t}\{x_j \geq 2\} > P_{k,t}\{x_i \geq 1\}$;
$P_{fulfill}$	The probability of having at least a *missing number*$(N_e - \lvert R_k \rvert)$ of *potential returners* placing another call before the end of cycle;
$P^*_{fulfill}$	The probability of having at least a *missing number*$(N_e - \lvert R_k \rvert)$ of sensed results returned from *potential returners* and *future-surer candidates* $((A_k - R_k) \cup FS_{U_i})$;

5.2. Parameter Estimation using Accumulated Traces

According to the Poisson law and Maximum Likelihood Estimation (MLE) [Gourieroux et al. 1984], when $k \leq M$, the Poisson intensity $\lambda_{i,k,t} = C_{i,k,t}$ refers to the number of calls made/received by U_i from the start of cycle k to time t; when $k > M$, $\lambda_{i,k,t}$ 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:

$$\lambda_{i,k,t} = \frac{\sum_{1 \leq k' \leq \lfloor k/M \rfloor} C_{i,(k' * M + k \,\mathbf{mod}\, M)} + C_{i,k,t}}{\lfloor k/M \rfloor}, \tag{1}$$

where $C_{i,(k' * M + k \,\mathbf{mod}\, M)}$ $(1 \leq k' \leq \lfloor k/M \rfloor)$ refers to the number of phone calls made/received by U_i in all previous *corresponding cycles* of cycle k (cycle k is included). For example, as shown in Figure 3, the sensing cycle k is from 10:00 to 12:00 in the fourth day of the MCS task. Then, $C_{i,k \,\mathbf{mod}\, M} = 2$, $C_{i,M+k \,\mathbf{mod}\, M} = 3$ and $C_{i,2M+k \,\mathbf{mod}\, M} = 2$ stand for the numbers of phone calls made/received by U_i during the corresponding cycles in the first, second, and third days, respectively. Because only one phone call has been made/received by U_i from the start of cycle k to the elapsed time t, *EEMC* counts the number of phone calls made in current cycle as $C_{i,k,t} = 1$. Thus, in this example, the Poisson intensity of U_i in the sensing cycle k is estimated to be $\lambda_{i,k,t} = \frac{(2+3+2)+1}{4} = 2$.

6. ADAPTIVE PACE CONTROLLER FOR TASK ASSIGNMENT

In this section, we would like to introduce: (i) the *adaptive pace control mechanism* for task assignment, (ii) the probability estimation used in *adaptive pace control*

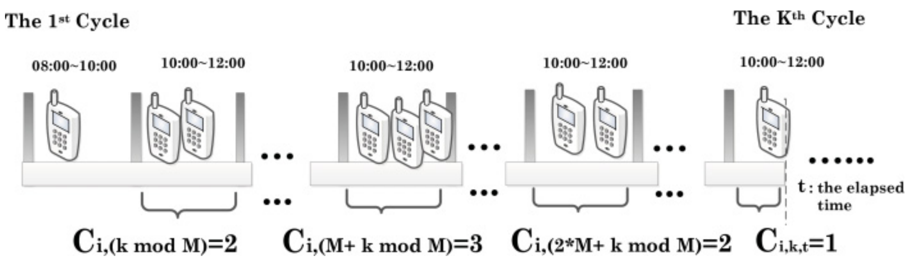


Fig. 3. An example: Estimating the parameter with U_i's accumulated call traces.

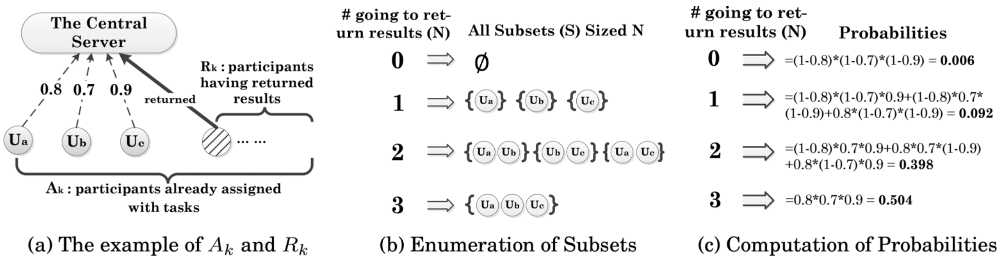


Fig. 4. The example of $P\{X_{k,t}(A_k - R_k) = N\}$ computing (best viewed in digital format).

mechanism (i.e., estimating if the *missing number* of sensed results can be returned from *potential returners*), and (iii) a *low-complexity algorithm* to reduce the time consumption of the probability estimation in the Adaptive Pace Controller.

6.1. Adaptive Pace Control for Task Assignment

Given the set 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.*

With $P_{fulfill}$ defined and the success probability threshold P_s given, *EEMC* controls the task assignment in a straightforward way—if $P_{fulfill} \geq P_s$ then further task assignments are not needed immediately, and *EEMC* stops making further task assignments; if $P_{fulfill} < P_s$ then further task assignments are still needed, and *EEMC* moves to the next step for task assignment: decision making (please see also in the pseudo code between Lines 9–11 of Algorithm 1). In this way, the key is to calculate $P_{fulfill}$.

6.2. Probability Estimation for Adaptive Pace Control

In order to estimate $P_{fulfill}$, we first define $P\{X_{k,t}(A_k - R_k) = N\}$ 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 $N \leq |A_k - R_k|$. To calculate this probability, we need to first enumerate all possible subsets of N participants from $A_k - R_k$. For each subset of N participants, we need to calculate the probability of having N participants placing at least another single call before the end of current cycle. Finally, as with the example shown in Figure 4, $P\{X_{k,t}(A_k - R_k) = N\}$ provides an estimation of the sum of probabilities for all possible subsets, and it is calculated as specified in Equation (2).

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

In this way, $P_{fulfill}$ is estimated as the sum of $P\{X_{k,t}(A_k - R_k) = N\}$, 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 Equation (3)).

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

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 predefined number of sensed results, thus $P_{fulfill} = 0$.

6.3. A Low-Complexity Algorithm for Adaptive Pace Controller Implementation

As the computation complexity of enumerating all subsets from a *n-length* set is $O(2^n)$, it is very time-consuming to solve Equation (2) through a subset-enumeration algorithm. For example, there are $2^{50} = 1.126 \times e^{15}$ subsets in a set with 50 elements. To reduce the computation complexity of $P_{fulfill}$ in Equation (2), we proposes an algorithm with $O(n^2)$ complexity. According to the Probability Generating Function Theory [Gardiner et al. 1985], $P\{X_{k,t}(A_k - R_k) = N\}$ is **equivalent** to the coefficient of z^N in the following polynomial over z:

$$\prod_{U_m \in A_k - R_k} (z * P_{k,t}\{x_m \geq 1\} + (1 - P_{k,t}\{x_m \geq 1\})). \tag{4}$$

Finally, we can resolve the given polynomials and calculate all necessary coefficients by using Algorithm 2.

7. NEAR-OPTIMAL DECISION MAKER FOR TASK ASSIGNMENT

Given the incoming call from one of the MCS participants and previous call records, the key algorithms of this step include (i) identifying all *future-surer candidates*, (ii) estimating if the *missing number* of sensed results can be returned from *future-surer candidates* and *potential returners*, and (iii) the Near-Optimal task assignment decision making.

7.1. 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 hasn't placed any call in the current cycle, i.e., $U_m \in S_1 \cup S_2 \cdots \cup S_{k-1} - S_{k,t}$, **and**
—U_m has a higher probability of placing at least two calls than U_i placing at least another call (i.e., $P_{k,t}\{x_m \geq 2\} > P_{k,t}\{x_i \geq 1\}$).

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

ALGORITHM 2: Computing Coefficients

```
Input: $A_k$, $R_k$, and $P_{k,t}\{x_m = n\}$
Output: coeffs– the array of coefficients
begin
    /* initiate the coefficients of polynomial.                    */
    coeffs ← NEW_ARRAY_OF_SIZE(1);
    coeffs[0] ← 1;
    /* Cumulative Product of Binomials                             */
    for $0 \le m < |A_k - R_k|$ do
        new_length ← LENGTH_OF(coeffs)+1;
        new_coeffs ← NEW_ARRAY_OF_SIZE(new_length);
        for $0 \le i <$ LENGTH_OF(coeffs) do
            new_coeffs[i] += coeffs[i] * (1-$P_{k,t}\{x_m \ge 1\}$);
            new_coeffs[i+1] += coeffs[i] * $P_{k,t}\{x_m \ge 1\}$;
        end
        coeffs ← new_coeffs;
    end
    return coeffs;
end
```

ALGORITHM 3: Identifying Future-Surer Candidates

```
Input: $S_1, S_2 \ldots, S_{k-1}, S_{k,t}$ and $U_i$
Output: $FS_{U_i}$ : the set of future-surer users for $U_i$
$FS_{U_i} \leftarrow \emptyset$;
for $U_l \in S_1 \cup S_2 \cdots \cup S_{k-1} - S_{k,t}$ do
    if $P_{k,t}\{x_l \ge 2\} > P_{k,t}\{x_i \ge 1\}$ then $FS_{U_i} \cup \{U_l\} \rightarrow FS_{U_i}$
end
return $FS_{U_i}$;
```

7.2. 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 Equation (5):

$$P'_{k,t}(U_m) = \begin{cases} P_{k,t}\{x_m \ge 1\}, & U_m \in (A_k - R_k). \\ P_{k,t}\{x_m \ge 2\}, & U_m \in FS_{U_i} \end{cases} \tag{5}$$

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 Equation (3), $P^*_{fulfill}$ can be computed using Equations (6) and (7), where $\mathrm{P}\{X^*_{k,t}(FS_{U_i} \cup (A_k - R_k)) = N\}$ refers to the probability of N sensed results being returned from *future-surer candidates* and *potential returners*.

$$P^*_{fulfill} = \begin{cases} 0 & , |A_k \cup FS_{U_i}| < N_e \\ \sum\limits_{N \geq N_e - |R_k|}^{N \leq |(A_k - R_k) \cup FS_{U_i}|} P\{X^*_{k,t}(FS_{U_i} \cup (A_k - R_k)) = N\}, & |A_k \cup FS_{U_i}| \geq N_e \end{cases} \tag{6}$$

$$\begin{aligned} P\{X^*_{k,t}(FS_{U_i} \cup (A_k - R_k)) = N\} &= \sum_{\forall s \subset (A_k - R_k) \cup FS_{U_i}}^{|s|=N} \prod_{\forall U_m \in s} P'_{k,t}(U_m) \\ &\quad \times \prod_{\forall U_m \notin s} (1 - P'_{k,t}(U_m)) \end{aligned} \tag{7}$$

Similar to Equation (2), $P\{X^*_{k,t}(FS_{U_i} \cup (A_k - R_k)) = N\}$ is **equivalent** to the coefficient of z^N in polynomial:

$$\prod_{U_m \in (A_k - R_k) \cup FS_{U_i}} (z * P'_{k,t}(U_m) + 1 - P'_{k,t}(U_m)). \tag{8}$$

Obviously, all coefficients in Equation (8) can be resolved by an algorithm similar to Algorithm 2 under $O(n^2)$ complexity.

7.3. Near-Optimal Task Assignment Decision Making

With $P^*_{fulfill}$ computed and the threshold P_s, *EEMC* assigns a task to the current caller (U_i) if $P^*_{fulfill}$ is lower than P^*_s. The pseudo code of Near-Optimal task assignment decision making is shown in Lines 12–28 of Algorithm 1.

Please note that, according to our proposed *Future-surer Candidates Identification* listed in Algorithm 3, it is impossible to discover any *future-surer candidates* in the sensing cycles of the first day in an MCS task (i.e., $k \leq M$). Thus, there needs a method to ***cold-start*** the proposed *Near-Optimal Decision Maker* in the first day of an MCS task. Rather than comparing the current caller with potential users in the future, we propose a method to make the task assignment decision making based on the *current caller's* next-call probability alone. As shown in Lines 12–19, when $k \leq M$, this step decides to assign a task to the current caller U_i, if $P_{k,t}\{x_i \geq 1\} > P_{k,t}\{x_i = 0\}$. If U_i doesn't have a higher probability of placing another call before the end of cycle, then this step skips the current caller.

8. EXPERIMENTAL SETUPS

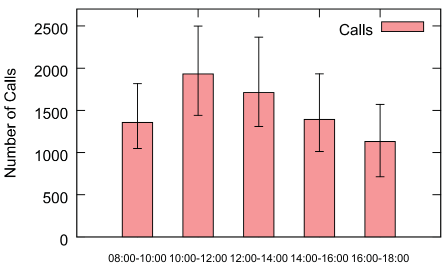
In this section, we introduce two baselines for comparison with *EEMC*, then present an overview of our dataset and experiment configuration.

8.1. Baseline Methods and Parameter Settings

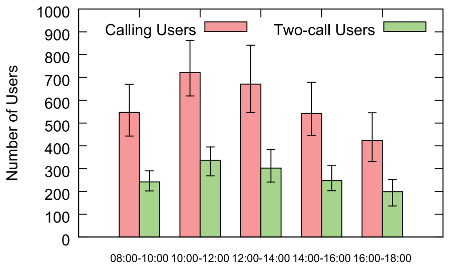
In this section, we present the configurations and setups of our proposed baselines.

—***Greedy:*** The most obvious method for task assignment to ensure a predefined number of sensed results is the ***Greedy method***, which assigns the sensing task to each new calling participant until the expected number of sensed results are returned (i.e., until $|R_k| = N_e$). This baseline method provides an upper bound of total task assignments to ensure that the expected number of participants return data.

—***Pace:*** Because there is a delay between task assignment to a participant and the return of the sensed result from the participant (through making another call), redundant tasks could be assigned when the expected number of results have been returned. Indeed, if the expected number of returned results can be predicted in



(a) Number of Calls



(b) Number of Calling Users and Two-call Users

Fig. 5. Statistics of evaluation traces in D4D dataset.

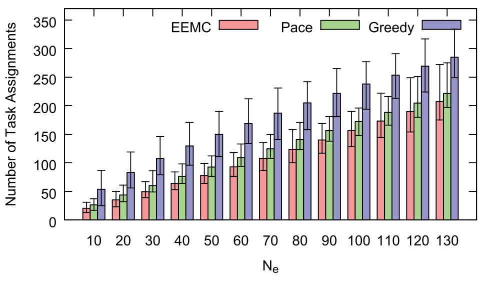
advance, based on our proposed *Adaptive Pace Controller* module, the task assignment process could terminate earlier to avoid some unnecessary task assignment. The task assignment strategy leveraging the *adaptive pace controller for task assignment* module is defined as ***Pace controller-based method*** (or ***Pace*** in short).

The comparison between **Greedy** and **Pace** shows whether our proposed *Pace controller* can stop making further task assignments when the tasks already assigned are sufficient to guarantee the expected number of participants returning. Furthermore, compared to the Pace method, *EEMC* assigns tasks considering not only participants with tasks already assigned, but also the future callers/receivers. Thus, the comparison between **Pace** and *EEMC* demonstrates the improved performance of our proposed *Optimal Task Assignment Decision Making* method with respect to the minimization of the total number of task assignments. In all experiments, we set the threshold $P_s = 99.99\%$ for the evaluation of *Pace controller-based* baseline and *EEMC*.

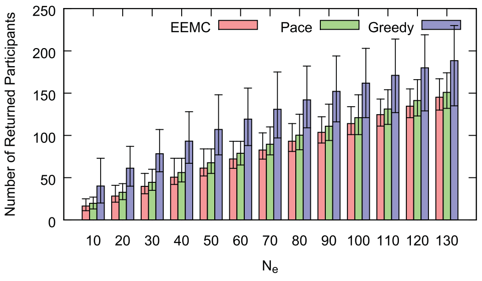
8.2. Dataset and Experiment Setups

The D4D project collected 4 months of Call Detail Records (CDR) from Orange Telecom subscribers in the Ivory Coast, nationwide. Each CDR record includes the calling time, the cellular tower where the call was made/received, and the identifier of the mobile phone user. The D4D dataset has been split into consecutive 2-week periods. In each time period, 50,000 users are *randomly selected* from all subscribers in the Ivory Coast. All selected users are assigned anonymized identifiers. Thus, in this study, we assume that each MCS task lasts for 2 weeks. For each participant, we can retrieve her call traces in the current MCS task but cannot link to her previous records. As we discussed in Section 1, the mobile phone users inside the D4D dataset perfectly satisfy the privacy constraints for MCS participants. The detailed experiment settings are as follow:

(1) *Sensing Cycles:* We evaluate *EEMC* when monitoring the CBD of Abidjan (shown in Figure 1(a)) from Monday to Friday every week (holidays excluded). Each sensing cycle lasts 2 hours, and we sense only in the working hours from 08:00 to 18:00 of a day. Thus, we split a working day into five equal-length sensing cycles (i.e. 8:00–10:00, . . . ,16:00–18:00).
(2) *Participants:* In every 2-week period, 2,000–3,000 mobile phone users recorded in our dataset would place phone calls in the target area (i.e., approximately 0.3% local mobile subscribers living in the target area). We assume them to be participants in our MCS task. To further introduce the call behaviors of these participants, we count the numbers of phone calls, calling participants, and frequent users (those with two or more phone calls in a sensing cycle). The average/minimum/maximum numbers of these are shown in Figure 5. It shows that (i) on average, 1,200–2,000 calls will be received/made in the target region per sensing cycle, (ii) on average, no



(a) Task Assignments



(b) Returned Participants

Fig. 6. Comparison of task assignments and returned participants: EEMC vs. Pace vs. Greedy.

more than half the calling participants (i.e., approximately 200 participants) will place another call in a sensing cycle, and (iii) at least 136 users will place two or more phone calls in a sensing cycle.

(3) *The Expected Number of Sensed Results:* Consequently, we cannot ensure the expected number of participants returning in each of sensing cycles, if we expect more than 136 participants to return. Thus, for our experiments, we set the expected number of returned participants in each cycle N_e to be evenly distributed from 10 to 130 (i.e., $N_e = 10, 20, 30, \ldots 130$).

In the following sections, we introduce the evaluation results based on the experiment setups just specified.

9. EVALUATION RESULTS

In this section, we present and compare the evaluation results of *EEMC*, Pace, and Greedy methods:

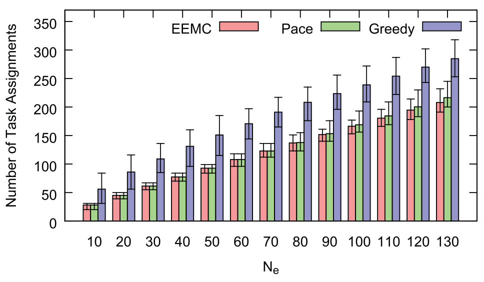
(1) In Section 9.1, we show the overall performance comparison of *EEMC*, Pace, and Greedy, including the average/maximal/minimal number of task assignments and returned participants in each sensing cycle.
(2) In Section 9.2, we extract and present the performance of *EEMC* at the cold start period.
(3) In Section 9.3, we examine in detail the execution of the three algorithms on a subset of the experimental data in order to illustrate their behaviors. Through a case study of *EEMC*, Pace, and Greedy, we analyze how *EEMC* assigns tasks step by step in a sensing cycle.
(4) In Section 9.4, we estimate how much energy our proposed *EEMC* scheme can save in data transfer compared to the commonly seen 3G-based MCS schemes.

The results will combine to show the excellence of *EEMC* with respect to minimizing the total number of task assignments and saving overall energy consumption while guaranteeing the expected number of participants returning results.

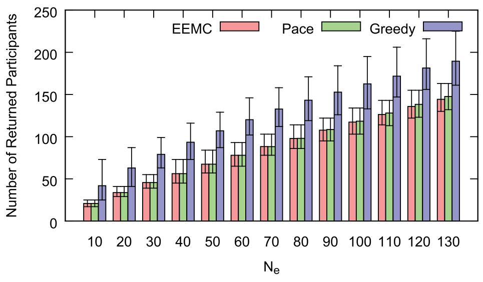
9.1. Performance Comparison

In Figure 6, we present the average/minimal/maximal numbers of task assignments and returned participants for *EEMC*, Pace, and Greedy in each sensing cycle with varied N_e (10 to 130).

(1) **Number of Returned Participants.** The primary constraint of our work is to ensure the expected number of participants returning their sensing results. Figure 6(b) shows that, for either *EEMC*, Pace, or Greedy, the minimal number of



(a) Task Assignments



(b) Returned Participants

Fig. 7. Number of task assignments and returned participants in cold start period.

returned participants in each sensing cycle is equal to or greater than the expected number (N_e). It means, with any of these methods, the MCS tasks can be successfully fulfilled in each of sensing cycles. 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 *EEMC* is 3.8%, whichi is 17% less than Pace and 23–59% less than Greedy on average.

(2) **Number of Task Assignments.** Furthermore, the optimization goal of *EEMC* is to minimize the total number of task assignments. Figure 6(a) shows clearly that *EEMC* assigns less tasks to participants than Pace and Greedy. On average, *EEMC* reduces task assignments by 6–23% when compared to Pace, and it reduces task assignments by 27–62% when compared to the Greedy method.

For the Greedy method, it is obvious that the delay between the task assignment to the participant (*who returns the N_e^{th} sensed result in this cycle*) and the return of the sensed result cause a large number of redundant task assignments and unnecessary returned results, whereas the Pace method may assign tasks to the participants not placing another call in the sensing cycle, which leads to high redundant task assignments. In contrast, for *EEMC*, the reason for the redundant task assignment is mainly due to the inaccurate call prediction with a limited number of call traces. However, in terms of the number of task assignments and returned results, *EEMC* still outperforms all other methods in all conditions. In summary, we can conclude that the overall performance of *EEMC* is the best among the three schemes. It ensures data collection from the expected number (10–130) of participants and assigns the minimal number of redundant tasks among all evaluated schemes.

9.2. Cold-Start Performance

As discussed in Section 7.3, *EEMC* needs to cold-start its Near-Optimal decision-making module in the first day of every MCS task (namely *cold-start periods*). Figure 7(a) illustrates the number of task assignments per cycle in the cold-start periods, whereas Figure 7(b) presents the number of returned participants. During the cold-start periods, *EEMC* slightly outperforms Pace but performs worse than the average in normal periods. This is because the Near-Optimal decision-making module assigns tasks to callers with “*maximal probabilities*” to return their sensing results after the cold-start period. Pace also performs worse during the cold-start periods due to the inaccuracy of probability estimation at the beginning of MCS tasks.

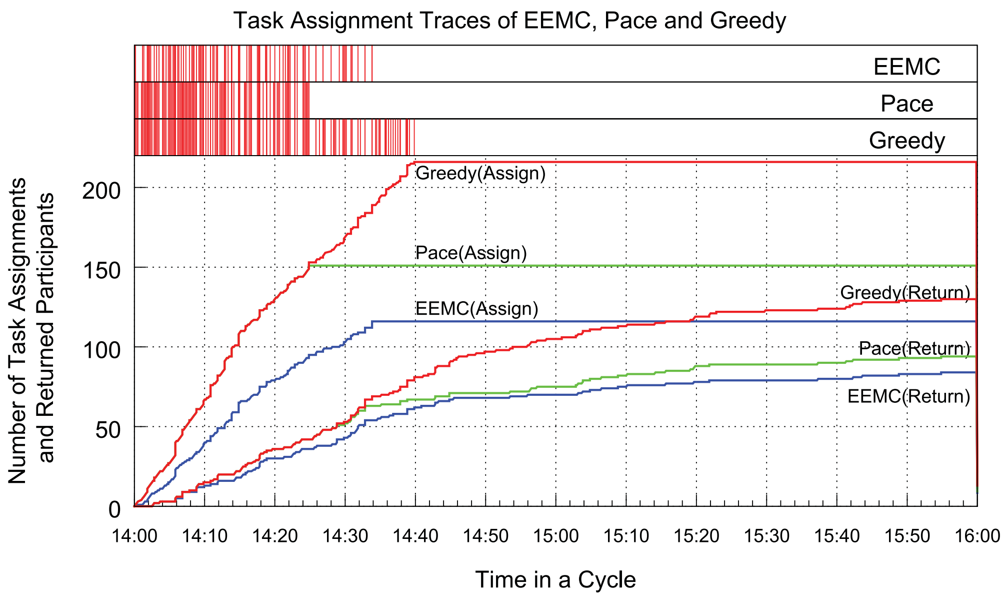


Fig. 8. Number of task assignments and returned participants varying with time in the cycle of 10:00−12:00, December 15, 2011.

9.3. Case Study and Analysis

To verify whether each proposed algorithm works as designed using the real-world datasets, we investigate how *EEMC* assigns tasks inside a single (typical) sensing cycle. We choose the sensing cycle of 14:00−16:00, December 15, 2011, for the case study and set the expected number of returned participant as 80 (i.e., $N_e = 80$). Please note that this sensing task is not in the cold-start period.

In Figure 8, we count the number of task assignments and returned participants varying against time inside the chosen sensing cycle and visualize the process of task assignments. We evaluate all three schemes, observing that:

—Comparing Greedy with Pace, Pace assigns tasks to the same calling participants as Greedy but stops assigning new tasks at 14:24 when 42 participants return their sensed results, whereas Greedy keeps assigning new tasks until 14:39, when a total of 80 participants return their sensed results. The Pace method stops 15 minutes earlier than the Greedy method, which causes 65 fewer redundant task assignments and 36 fewer unnecessary returned results. Such improvement is contributed to by our proposed *Adaptive Pace Controller* that stops assigning a new task when it estimates that the tasks already assigned are enough to fulfill the minimum requirement.

—Comparing *EEMC* with Pace, *EEMC* gives up assigning tasks to calling participants even in the beginning of the cycle because it predicts there are a sufficient number of future users who have higher probabilities to place two calls before the end of the current cycle. We can see that *EEMC* holds the tasks and leaves them to *future-surer users*. In this way, *EEMC* stops making new task assignments later (at 14:33, when 54 participants return) but assigns fewer tasks (35 fewer) than the Pace to fulfill the task. Since *EEMC* always chooses the users with higher probabilities to place two calls, it can guarantee the expected number of participants returning after assigning a smaller number of tasks.

Table III. Data Transfer Energy Consumption Estimation

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

Table IV. Energy Consumption Comparison: 3G-based vs Parallel+3G-based (P+3G) vs *EEMC* vs Pace vs Greedy

N_e	3G (J)	P+3G (J)	*EEMC (J)*	Pace (J)	Greedy(J)
10	240	150	110.37	138.00	**281.48**
20	480	300	190.18	229.32	**433.75**
30	720	450	268.15	313.75	**557.88**
40	960	600	343.77	397.35	**668.28**
50	1200	750	417.66	480.78	**771.35**
60	1440	900	494.98	563.03	863.82
70	1680	1050	571.48	642.29	953.82
80	1920	1200	650.74	722.37	1040.85
90	2160	1350	730.73	801.59	1120.88
100	2400	1500	811.95	879.31	1199.57
110	2640	1650	893.13	958.64	1274.27
120	2880	1800	972.88	1037.97	1347.80
130	3120	1950	1057.31	1116.76	1419.49

Our analysis suggests that it is reasonable to conclude that all three algorithms in our comparison work as designed on the real-world datasets.

9.4. Energy Conservation Comparison

With the number of task assignments and returned results obtained, it becomes possible to estimate the energy consumption of *EEMC* and corresponding baselines. In this section, we would like to estimate how much energy our proposed *EEMC* scheme can save in data transfer compared to the following schemes:

—***3G-based Scheme***: Receives the task assignment by establishing a new 3G connection, and returns the sensed results by establishing another 3G connection.
—***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.

These two schemes do not need redundant task assignments (i.e., both methods can secure N_e participants returning their sensed results through assigning tasks to N_e participants) since all the participants can return the sensed results via a new 3G connection by using these two schemes. Table III lists the overall energy consumption estimation formulas in data transfer for all the schemes; these formulas are based on:

(1) the common observations reported by existing literature [Balasubramanian et al. 2009; Nurminen 2010; Thiagarajan et al. 2012; R. Pease 2013] measuring the energy consumption of N95 and Android phones, and
(2) the assumption that the data packets for task assignment or sensed results are small (<10KB 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 purposes.

Table IV shows each scheme's average energy consumption per sensing cycle as N_e varies. *EEMC* outperforms all the other schemes. Specifically, it can save 54%–66%

energy compared to the 3G-based scheme; it can save 26%–46% energy compared to the Parallel+3G-based scheme. Note that these evaluations are based on a small number of expected sensed results (i.e., $N_e \leq 130$). If an MCS task needs more participants to collect sensed data, and there are more sensing cycles per day, the total energy saving will be much more significant. Interestingly, if we compare *EEMC*, Pace, and Greedy with the Parallel+3G-based scheme, we can see that *EEMC* outperforms all the other schemes in all the conditions, but the Greedy method consumes more energy than the Parallel+3G-based scheme when $N_e < 60$. In summary, all the evaluation results show the effectiveness of *EEMC* in saving energy consumption in data transfer for both individual participants and the whole crowds.

10. CONCLUSION AND DISCUSSION

In this article, we have presented *EEMC*, a framework to enable energy-efficient mobile crowdsensing, where the goal is to reduce energy consumption in data transfer for both individual participants and the whole crowd while securing the sensed result collection from a minimum number of participants within a specific timeframe (namely, a *sensing cycle*). The proposed framework embeds several mechanisms from existing work such as *parallel transfer* and *cycle-based delay-tolerant participatory sensing* into a novel *two-call-based MCS data transfer scheme*, which is capable of reducing energy consumption in data transfer for individual devices by 75% compared to the common 3G-based schemes. To reduce overall energy consumption for the whole crowd, we propose a two-step task assignment decision-making algorithm to avoid redundant task assignments. Evaluations with a large-scale real-world dataset show that the proposed algorithm constantly outperforms baseline approaches in terms of task assignment, and *EEMC* can reduce overall energy consumption in data transfer by 54%–66% when compared to the 3G-based schemes.

Due to space constraints, there are still several problems that are not fully elaborated or addressed in this work. For example:

—**Using Advanced Predictor and Adaptive Threshold:** The effectiveness of *EEMC* in reducing the *overall energy consumption* (e.g., the number of redundant task assignment and returned participants) relies on the precision of the adopted call predictor and the threshold settings (i.e., the settings of P_s) used in the Pace Controller/Decision Maker modules. In this study, we simply use the inhomogeneous Poisson process to model/predict future calls, and we choose a fixed threshold in all conditions. In future work, we plan to improve *EEMC* by adopting an advanced call predictor and study the novel Pace Controller/Decision Maker modules leveraging of self-adaptive threshold settings.

—**Integrating with the Telecom Operator:** In this work, we assume that, in collaboration with the telecom operator, the *EEMC* smartphone client could piggyback MCS data over participant's calls while the *EEMC* server could access each participant's call traces. To piggyback MCS data over cell phone calls, each participant's *EEMC* client should be able to access a 3G/LTE network (e.g., WCDMA), where the parallel connection [Nurminen 2010] is enabled by the telecom operator. In order to access the mobile call traces, there thus needs to be a method to integrate the *EEMC* server with the telecom operator. Please refer to our previous work [Xiong et al. 2013b], where we studied an alternative to integrating an *EEMC* server with the telecom operator.

—**Enabling Coverage-Constrained Sensing:** In this research, we have not proposed any techniques to tackle the coverage issue in mobile crowdsensing. Instead, we focus on the number of sensed results returned from a single target area. Assuming the framework is capable of collecting a predefined number of sensed results from each subarea [Reddy et al. 2010; Sheng et al. 2012] or a certain percentage of subareas [Ahmed et al. 2011; Hachem et al. 2013] in the whole target region, then our proposed framework can be adopted to enable *coverage-constrained* crowdsensing applications by dividing the target region into multiple subareas and using multiple *EEMC* servers to collect sensed results in each subarea collaboratively.

—**Leveraging Additional Energy-Saving Techniques:** Apart from piggybacking MCS data over 3G calls, other data transfer methods (e.g., piggybacking data over 3G data connection or transferring data via WiFi) also consume less energy when compared to common 3G-based solutions. Furthermore, 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 leveraging multiple energy-saving strategies to further reduce energy consumption in a holistic manner.

—**Handling the *Corner* Cases:** Inside an overall target area, there might exist some subareas, namely *corners*, where few or no cell towers have been installed. In order to get sensed results from the whole target area including corners, we could use *EEMC* complemented by many existing sensing approaches (e.g., deploying static sensor infrastructure in corners). In future work, we will study the methods integrating *EEMC* with existing sensor infrastructure.

—**Energy Reduction of *EEMC*:** In our work, we attempt to reduce energy consumption caused by MCS applications from two aspects: (i) the energy consumed by the MCS application on individual mobile phone and (ii) the overall energy consumption caused by the whole MCS task on all mobile phones. In terms of reducing individual energy consumption, when compared to common MCS alternatives (e.g., 3G-based MCS), *EEMC* consumes less energy in data transfer but without causing extra energy consumption in computation and sensing. In terms of overall energy consumption, *EEMC* intends to assign as few MCS tasks as possible in order to reduce the overall energy consumption on the top of *two-call-based data transfer* mechanism. In our future work, we plan to profile the energy consumption caused by realistic MCS applications using *EEMC*, and we will take the energy consumption of computation and sensing into account for task assignment decision making.

—**Fairness in Allocation of Tasks:** Users may be more motivated to join the sensing crowd if they know that energy resources are used fairly. In other words, that tasks are distributed as equally as possible among the crowdsensing members. They may also consider it unfair if they are allocated tasks when their mobile phone batteries are below a certain threshold value.

Considering these open issues, in our future work, we plan to broaden and deepen this research in several directions. First, we will attempt to design an optimal task assignment mechanism based on advanced call prediction methods and study the impact of threshold change on the task assignment, coverage, and number of returned participants. Second, we will study the coverage issue of mobile crowdsensing by considering the physical coverage of mobile phone sensors and the fine-grained mobility (e.g., GPS location) of participants. Third, we would like to develop a real mobile crowdsensing platform leveraging mobile phones of volunteers, thus enabling a series of urban-scale environment monitoring services.

REFERENCES

A. Ahmed, K. Yasumoto, Y. Yamauchi, and M. Ito. 2011. Distance and time based node selection for probabilistic coverage in people-centric sensing. In *Proceedings of the 2011 IEEE Conference on Sensing, Communication and Networking*. IEEE, 134–142.

D. Akimura, Y. Kawahara, and T. Asami. 2012. Compressed sensing method for human activity sensing using mobile phone accelerometers. In *Proceedings of the 2012 Networked Sensing Systems Conference*. IEEE, 1–4.

N. Balasubramanian, A. Balasubramanian, and A. Venkataramani. 2009. Energy consumption in mobile phones: A measurement study and implications for network applications. In *Proceedings of the 2009 ACM SIGCOMM Conference on Internet Measurement Conference*. ACM, 280–293.

V. D. Blondel, M. Esch, C. Chan, F. Clerot, P. Deville, E. Huens, F. Morlot, Z. Smoreda, and C. Ziemlicki. 2012. Data for development: The D4D challenge on mobile phone data. *CoRR* abs/1210.0137 (2012).

J. Brown, J. Finney, C. Efstratiou, B. Green, N. Davies, M. Lowton, and G. Kortuem. 2007. Network interrupts: Supporting delay sensitive applications in low power wireless control networks. In *Proceedings of the 2007 ACM Workshop on Challenged Networks*. ACM, 51–58.

Y. Chon, N. D. Lane, F. Li, H. Cha, and F. Zhao. 2012. Automatically characterizing places with opportunistic crowdsensing using smartphones. In *Proceedings of the 2012 International Conference on Ubiquitous Computing, ACM*.

D. Chu, N. D. Lane, T. T. Lai, C. Pang, X. Meng, Q. Guo, F. Li, and F. Zhao. 2011. Balancing energy, latency and accuracy for mobile sensor data classification. In *Proceedings of the 2011 ACM International Conference on Embedded Networked Sensor Systems*. ACM, 54–67.

G. Cohn, S. Gupta, T.-J. Lee, D. Morris, J. R. Smith, M. S. Reynolds, D. S. Tan, and S. N. Patel. 2012. An ultra-low-power human body motion sensor using static electric field sensing. In *Proceedings of the 2012 ACM International Conference on Ubiquitous Computing*.

P. Dutta, P. M. Aoki, N. Kumar, A. Mainwaring, C. Myers, W. Willett, and A. Woodruff. 2009. Common sense: Participatory urban sensing using a network of handheld air quality monitors. In *Proceedings of the 2009 ACM Conference on Embedded Networked Sensor Systems*. ACM, 349–350.

F. Ferrari, M. Zimmerling, L. Mottola, and L. Thiele. 2012. Low-power wireless bus. In *Proceedings of the 2012 ACM Conference on Embedded Networked Sensor Systems*. ACM, 1–14.

M. Ficek, N. Clark, and L. Kencl. 2012. Can crowdsensing beat dynamic cell-ID? In *Proceedings of the 2012 International Workshop on Sensing Applications on Mobile Phones*. ACM, 10.

K. Frank. 2011. *Adaptive and Tractable Bayesian Context Inference for Resource Constrained Devices*. Ph.D. Dissertation. Waterford Institute of Technology.

R. K. Ganti, F. Ye, and H. Lei. 2011. Mobile crowdsensing: Current state and future challenges. *IEEE Communications Magazine* 49 (2011), 32–39.

C. W. Gardiner et al. 1985. *Handbook of Stochastic Methods*. Vol. 3. Springer, Berlin.

D. Gordon, J. Czerny, T. Miyaki, and M. Beigl. 2012. Energy-efficient activity recognition using prediction. In *Proceedings of the 2012 International Symposium on Wearable Computers*. IEEE, 29–36.

C. Gourieroux, A. Monfort, and A. Trognon. 1984. Pseudo maximum likelihood methods: Applications to Poisson models. *Econometrica: Journal of the Econometric Society* (1984), 701–720.

Bin Guo, Zhiwen Yu, Daqing Zhang, and Xingshe Zhou. 2014. From participatory sensing to mobile crowd sensing. In *Proceedings of the 2014 IEEE Pervasive Computing and Communication Workshops*. IEEE, 593–598.

S. Hachem, A. Pathak, and V. Issarny. 2013. Probabilistic registration for large-scale mobile participatory sensing. In *Proceedings of the 2013 IEEE International Conference on Pervasive Computing and Communications*, Vol. 18. 22.

S. Isaacman, R. Becker, R. Cáceres, S. Kobourov, M. Martonosi, J. Rowland, and A. Varshavsky. 2011. Identifying important places in people?s lives from cellular network data. In *Proceedings of the 2011 International Conference on Pervasive Computing*. Springer, 133–151.

P. P. Jayaraman, A. Sinha, W. Sherchan, S. Krishnaswamy, A. Zaslavsky, P. D. Haghighi, S. Loke, and M. T. Do. 2012. Here-n-Now: A framework for context-aware mobile crowdsensing (Demo). In *Proceedings of the 2012 International Conference on Pervasive Computing*. Springer, 1–4.

Y. Jiang, D. Li, G. Yang, Q. Lv, and Z. Liu. 2011. Deliberation for intuition: A framework for energy-efficient trip detection on cellular phones. In *Proceedings of the 2011 ACM International Conference on Ubiquitous Computing*. ACM, 315–324.

M. B. Kjærgaard, S. Bhattacharya, H. Blunck, and P. Nurmi. 2011. Energy-efficient trajectory tracking for mobile devices. In *Proceedings of the 2011 ACM International Conference on Mobile Systems, Applications, and Services*. ACM, 307–320.

N. D. Lane, Y. Xu, H. Lu, S. Hu, T. Choudhury, A. T. Campbell, and F. Zhao. 2011. Enabling large-scale human activity inference on smartphones using community similarity networks (csn). In *Proceedings of the 2011 ACM International Conference on Ubiquitous Computing*. ACM, 355–364.

E. C. Larson, T. J. Lee, S. Liu, M. Rosenfeld, and S. N. Patel. 2011. Accurate and privacy preserving cough sensing using a low-cost microphone. In *Proceedings of the 2011 ACM International Conference on Ubiquitous Computing*. ACM, 375–384.

Y.-B. Lin. 1997. Reducing location update cost in a PCS network. *IEEE/ACM Transactions on Networking (TON)* 5, 1 (1997), 25–33.

S. Mathur, T. Jin, N. Kasturirangan, J. Chandrasekaran, W. Xue, M. Gruteser, and W. Trappe. 2010. Parknet: Drive-by sensing of road-side parking statistics. In *Proceedings of the 2010 International Conference on Mobile Systems, Applications, and Services*. ACM, 123–136.

M. Musolesi, M. Piraccini, K. Fodor, A. Corradi, and A. T. Campbell. 2010. Supporting energy-efficient uploading strategies for continuous sensing applications on mobile phones. *Pervasive Computing* (2010), 355–372.

J. K. Nurminen. 2010. Parallel connections and their effect on the battery consumption of a mobile phone. In *Proceedings of the 2010 IEEE Consumer Communications and Networking Conference*. IEEE, 1–5.

B. Pásztor, M. Musolesi, and C. Mascolo. 2007. Opportunistic mobile sensor data collection with scar. In *Proceedings of the 2007 IEEE Conference on Mobile Ad-hoc and Sensor Systems*. IEEE, 1–12.

D. Philipp, J. Stachowiak, P. Alt, F. Dürr, and K. Rothermel. 2013. DrOPS: Model-driven optimization for public sensing systems. In *Proceedings of the 2013 IEEE International Conference on Pervasive Computing and Communications*. IEEE, 185–192.

S. Phithakkitnukoon and R. Dantu. 2011. Towards ubiquitous computing with call prediction. *ACM SIGMOBILE Mobile Computing and Communications Review* 15, 1 (2011), 52–64.

S. Phithakkitnukoon, R. Dantu, R. Claxton, and N. Eagle. 2011. Behavior-based adaptive call predictor. *ACM Transactions on Autonomous and Adaptive Systems* 6, 3 (2011), 21.

D. Puccinelli and S. Giordano. 2013. Connectivity and energy usage in low-power wireless: An experimental study. In *Proceedings of PerCom Workshops*. IEEE, 590–595.

D. Puccinelli, S. Giordano, M. Zuniga, and P. J. Marrón. 2012. Broadcast-free collection protocol. In *Proceedings of the 2011 ACM International Conference on Embedded Networked Sensor Systems*. ACM, 29–42.

R. Pease. 2013. What is killing smartphones? BBC - Future - Technology. (2013). http://www.bbc.com/future/story/20130227-what-is-killing-smartphones.

M.-R. Ra, B. Liu, T. F. La Porta, and R. Govindan. 2012a. Medusa: A programming framework for crowd-sensing applications. In *Proceedings of the 2012 International Conference on Mobile Systems, Applications, and Services*. ACM, 337–350.

M.-R. Ra, B. Priyantha, A. Kansal, and J. Liu. 2012b. Improving energy efficiency of personal sensing applications with heterogeneous multi-processors. In *Proceedings of the 2012 ACM International Conference on Ubiquitous Computing*. ACM, 1–10.

K. K. Rachuri, C. Efstratiou, I. Leontiadis, C. Mascolo, and P. J. Rentfrow. 2013. METIS: Exploring mobile phone sensing offloading for efficiently supporting social sensing applications. In *Proceedings of the 2013 IEEE Conference on Pervasive Computing and Communications*. IEEE, 85–93.

K. K. Rachuri, C. Mascolo, M. Musolesi, and P. J. Rentfrow. 2011. Sociablesense: Exploring the trade-offs of adaptive sampling and computation offloading for social sensing. In *Proceedings of the 2011 Annual International Conference on Mobile Computing and Networking*. ACM, 73–84.

H. S. Ramos, T. Zhang, J. Liu, N. B. Priyantha, and A. Kansal. 2011. Leap: A low energy assisted GPS for trajectory-based services. *Proceedings of the 2011 ACM International Conference on Ubiquitous Computing* (2011), 335–344.

R. K. Rana, C. T. Chou, S. S. Kanhere, N. Bulusu, and W. Hu. 2010. Ear-phone: An end-to-end participatory urban noise mapping system. In *Proceedings of the 2010 ACM/IEEE International Conference on Information Processing in Sensor Networks*. ACM, 105–116.

S. Reddy, D. Estrin, and M. Srivastava. 2010. Recruitment framework for participatory sensing data collections. In *Proceedings of Pervasive*. 138–155.

X. Sheng, J. Tang, and W. Zhang. 2012. Energy-efficient collaborative sensing with mobile phones. In *Proceedings of the 2012 IEEE Conference on Computer Communications*. IEEE, 1916–1924.

W. Sherchan, P. P. Jayaraman, S. Krishnaswamy, A. Zaslavsky, S. Loke, and A. Sinha. 2012. Using on-the-move mining for mobile crowdsensing. In *Proceedings of the 2012 IEEE Conference on Mobile Data Management*. IEEE, 115–124.

J. Smith, A. Sample, P. Powledge, S. Roy, and A. Mamishev. 2006. A wirelessly-powered platform for sensing and computation. *Proceedings of the 2006 International Conference on Ubiquitous Computing* (2006), 495–506.

E. Soroush, K. Wu, and J. Pei. 2008. Fast and quality-guaranteed data streaming in resource-constrained sensor networks. In *Proceedings of the 2008 ACM International Symposium on Mobile Ad-hoc Networking and Computing*. ACM, 391–400.

A. Thiagarajan, L. Ravindranath, K. LaCurts, S. Madden, H. Balakrishnan, S. Toledo, and J. Eriksson. 2009. VTrack: Accurate, energy-aware road traffic delay estimation using mobile phones. In *Proceedings of the 2009 ACM Conference on Embedded Networked Sensor Systems*. ACM, 85–98.

N. Thiagarajan, G. Aggarwal, A. Nicoara, D. Boneh, and J. P. Singh. 2012. Who killed my battery?: Analyzing mobile browser energy consumption. In *Proceedings of the 2012 International Conference on World Wide Web*. ACM, 41–50.

L. Wang, D. Zhang, and H. Xiong. 2013. effSense: Energy-efficient and cost-effective data uploading in mobile crowdsensing. In *Proceedings of the 2013 ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication*. ACM, 1075–1086.

Y. Wang, J. Lin, M. Annavaram, Q. A. Jacobson, J. Hong, B. Krishnamachari, and N. Sadeh. 2009. A framework of energy efficient mobile sensing for automatic user state recognition. In *Proceedings of the 2009 International Conference on Mobile Systems, Applications, and Services*. ACM, 179–192.

J. Weinberg, L. D. Brown, and J. R. Stroud. 2007. Bayesian forecasting of an inhomogeneous Poisson process with applications to call center data. *Journal of the American Statististical Association* 102, 480 (2007), 1185–1198.

H. Weinschrott, F. Durr, and K. Rothermel. 2010. Streamshaper: Coordination algorithms for participatory mobile urban sensing. In *Proceedings of the 2010 IEEE Conference on Mobile Ad-hoc and Sensor Systems*. IEEE, 195–204.

H. Weinschrott, J. Weisser, F. Durr, and K. Rothermel. 2011. Participatory sensing algorithms for mobile object discovery in urban areas. In *Proceedings of the 2011 IEEE International Conference on Pervasive Computing and Communications*. IEEE, 128–135.

Y. Xiao, P. Simoens, P. Pillai, K. Ha, and M. Satyanarayanan. 2013. Lowering the barriers to large-scale mobile crowdsensing. In *Proceedings of the 2013 International Workshops on Mobile Computing Systems and Applications*.

H. Xiong, L. Wang, and D. Zhang. 2013a. EEMC: An energy-efficient mobile crowdsensing mechanism by reusing call/SMS connections. In *Proceedings of the 2013 Conference on the Analysis of Mobile Phone Datasets*. MIT, 323–329.

H. Xiong, D. Zhang, D. Zhang, V. Gauthier, K. Yang, and M. Becker. 2013b. MPaaS: Mobility prediction as a service in telecom cloud. *Information Systems Frontiers* (2013), 1–17.

J. Zheng and L. M. Ni. 2012. An unsupervised framework for sensing individual and cluster behavior patterns from human mobile data. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*. ACM, 153–162.

Received January 2014; revised June 2014; accepted July 2014