effSense: Energy-Efficient and Cost-Effective Data Uploading in Mobile Crowdsensing

Leye Wang
CNRS SAMOVAR
Institut Mines-TELECOM /
TELECOM SudParis
France
leye.wang@telecom-sudparis.eu

Haoyi Xiong
CNRS SAMOVAR
Institut Mines-TELECOM /
TELECOM SudParis
France
haoyi.xiong@telecom-sudparis.eu

Daqing Zhang
CNRS SAMOVAR
Institut Mines-TELECOM /
TELECOM SudParis
France
daqing.zhang@telecom-sudparis.eu

http://dx.doi.org/10.1145/2494091.2499575

Abstract

Energy consumption and mobile data cost are two key factors affecting users' willingness to participate in crowdsensing tasks. While *data-plan users* are mostly concerned about the energy consumption, *non-data-plan* users are more sensitive to data transmission cost incurred. Traditional ways of data collection in mobile crowdsensing often go to two extremes: either uploading the sensed data online in real-time or fully offline after the whole sensing task is finished. In this paper, we propose effSense - a novel energy-efficient and cost-effective data uploading framework leveraging the *delay-tolerant* mechanisms. Specifically, effSense reduces the data cost of *non-data-plan users* by maximally offloading the data to Bluetooth/WiFi gateways or data-plan users encountered to relay the data to the server; it reduces energy consumption of *data-plan users* by uploading data in parallel with a call or using less-energy demand networks (e.g. Bluetooth). By leveraging the prediction of critical events such as user's future calls or encounters, effSense selects the optimal uploading scheme for both types of users. Our evaluation with MIT Reality Mining and Nodobo datasets show that effSense can save 55% $\sim$ 65% energy and 45% $\sim$ 50% data cost for the two types of users, respectively, compared with the traditional uploading schemes.

Author Keywords
mobile crowdsensing, energy saving, mobile data usage

ACM Classification Keywords
C.2.1 [Network Architecture and Design]: Wireless Communications.

Introduction
As sensor equipped smartphones are getting more and more popular nowadays, mobile crowdsensing is becoming an effective way to carry out various sensing tasks such as environmental monitoring [14] and social sensing [13]. To improve user's experience in participating mobile crowdsensing tasks, how to minimize inconvenience incurred for users is a practical issue. In this regard, energy consumption and mobile data cost are two critical concerns. While energy consumption is related to the phone's battery life, mobile data cost is often associated with the fees paid, especially for the users who do not hold an unlimited data plan. Therefore, reducing energy consumption and data cost will hopefully encourage more people to actively participate in crowdsensing tasks.

Recent years have witnessed an increasing interest in improving user's experience in mobile crowdsensing, mainly from the perspective of energy saving. The proposed solutions include using dynamic sensing duty cycle [13], making trade-off between local and remote computation [13, 16], reducing data uploading frequency by predicting missing data at server side [8], splitting the task smartly among appropriate users [15], etc.. These works assume that sensed data should be sent to a central server as soon as they are produced.

In fact, some mobile crowdsensing tasks do not necessarily require the sensed data to be uploaded in real-time. For example, in MIT Reality Mining project [5], the participant's mobile traces were collected to understand user's interests, activity patterns, etc. mainly for further analysis. Actually, the MIT Reality Mining project collected users' data in two ways. (1) 30 participants were provided with a mobile data plan[4]. These users uploaded their sensed data every night. (2) The other participants' data was stored in the SD-cards and was collected after the mobile phones were returned in the end of the project. For both types of the participants, certain time of delay between sensing and uploading is allowed.

Thus, in this paper, for the crowdsensing task that does not require real-time sensed data uploading (called a *delay-tolerant crowdsensnig task*), we attempt to design a novel data uploading framework (named as *effSense*) leveraging heterogeneous networks (e.g. 3G, WiFi and Bluetooth), in order to enable (1) users without data plan (called *non-data-plan users*) to reduce data transmission cost by relaying data to bluetooth gateway or other mobile phones encountered (rather than via 3G network), (2) users with data plan (called *data-plan users*) to reduce the energy consumption in data uploading.

effSense is designed based on the following observations:

1. Non-data-plan users can eliminate mobile data cost in data uploading by using zero-cost network such as Bluetooth and WiFi. For example, they can upload data to the server directly via WiFi, or transfer data to another device via Bluetooth if the other device can relay data to the server without incurring extra cost.

2. Data-plan users can reduce energy consumption in data uploading via the energy-efficient methods other than establishing a new 3G connection. For example, coupling a 3G data uploading task with a 3G voice call can save

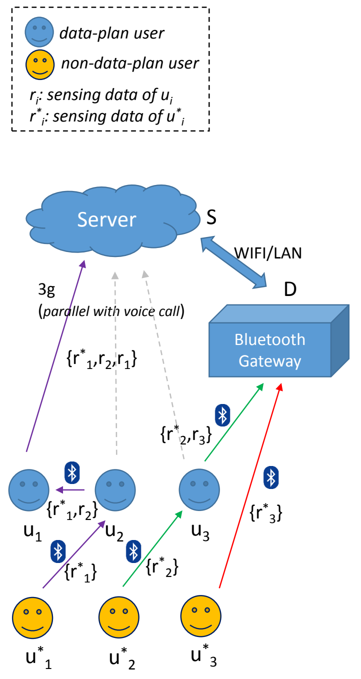


Figure 1: A running example of effSense

75% $\sim$ 90% energy cost [9]. Alternatively, uploading data via WiFi or Bluetooth consumes less energy than via normal 3G.

A running example: Figure 1 shows a simple example to illustrate the basic idea of effSense. Here, u_1, u_2, and u_3 are three data-plan users, while u_1^*, u_2^*, and u_3^* are three non-data-plan users. In addition, there is a server (S) and a fixed-location Bluetooth gateway device (D). Instead of uploading the sensed data directly via a specific 3G link, effSense might suggest the following data uploading paths to save data cost and energy consumption:

Path 1 (red): $u_3^* \rightarrow D \rightarrow S$. A non-data-plan user (u_3^*) uploads data via a Bluetooth gateway.

Path 2 (green): $u_2^* \rightarrow u_3 \rightarrow D \rightarrow S$. A non-data-plan user (u_2^*) relays data to a data-plan user (u_3) first and then u_3 uploads data via a Bluetooth gateway other than direct 3G to save both data cost and energy consumption.

Path 3 (purple): $u_1^* \rightarrow u_2 \rightarrow u_1 \rightarrow S$. The primary difference between this path and the two above is that a data-plan user (u_1) coupled the 3G data uploading with a voice call in the end, so as to save energy consumption.

As shown in this example, non-data-plan users are guided to upload their sensed data without mobile data cost, while data-plan users are recommended to upload data via energy-efficient methods other than direct 3G. The design objective of effSense is to ensure both data-plan and non-data-plan users upload their data to server within a predefined time line and with minimal energy consumption and data cost.

The key issues involved in designing effSense include:

1. Identify and predict the **critical events**. Typical critical events include *making a new call*, *meeting another user*, *encountering a Bluetooth gateway*, *connecting to a WiFi AP*, etc. Predicting the critical events for each user is the basis for effSense to select the right data relaying strategy and accurate prediction would result in better user's experience. Such predictions can be made by adopting existing activity prediction techniques [17, 11].

2. Estimate data uploading energy consumption associated to each *critical event*. Specifically, for data uploading, the energy consumption is not always proportional to the data size. Uploading several small-size sensed data together may require as much energy as uploading only one small-size sensed data. For instance, uploading data smaller than 10KB (whatever the exact size) via 3G consumes almost the same energy [1]. However, above a certain threshold, the energy for data uploading is roughly proportional to the data size.

3. Design real-time algorithms to decide whether data should be offloaded or kept at each individual event. The algorithms should be lightweight and executed on each phone locally without incurring much energy or data cost.

In summary, our paper has the following contributions:

1. To the best of our knowledge, this is the first work that aims to minimize both the energy consumption and mobile data cost in mobile crowdsensing by leveraging heterogeneous networks and delay-tolerant mechanisms.

2. We consider two types of users (non-data-plan and data-plan) with different goals and thus propose two data uploading schemes for each kind of users: one is pure greedy, the other is based on the mobility/call predictions. While the former one is quite effective in handling the cold-start problem when participants' historic call or

mobility logs are not available, the latter one can achieve better performance by leveraging the critical-event predictions according to the participants' historic logs.

3. We evaluate effSense with two datasets - MIT Reality Mining [5] and Nodobo [2]. The results show that effSense could upload about 45%∼50% of non-data-plan users' data without extra data cost, and reduce 55%∼65% of data-plan users' data-uploading energy consumption compared with the traditional method, under the assumption that data-plan users and non-data-plan users have the ratio of 3:4[*] and maximum delay is 24 hours.

Related Work

The literature related to effSense can be roughly divided into four categories as follows:

Mobile Crowdsensing Application Framework. To support various mobile crowdsensnig applications, many frameworks have been studied, such as task assignment [15] and data collection [8, 16]. Our work also provides a framework, while the primary target is different from the existing works as we optimize both non-data-plan and data-plan users' experience in terms of data uploading.

Energy Conservation in Mobile Sensing. The energy of a mobile device in a crowdsensing task is mainly consumed in three phases: *sensing*, *computing* and *data uploading*. While effSense does not target at the energy saving in the first two phases, the existing approaches in these two phases [7, 3] can be incorporated into effSense. To reduce the energy consumption in data uploading, while researchers have proposed various approaches such as adopting low-power communication protocols [12] and using relay nodes [6], effSense selects an adaptive set of these methods to reduce energy consumption.

Mobile Data Cost Conservation in Data Uploading. To save data cost in data uploading, existing works are focused on reducing data size by incurring additional computation on the local mobile device [13, 16]. While these existing approaches can reduce data cost largely when the size of raw data is big, they cannot eliminate it completely. As effSense targets to eliminate data cost for non-data-plan users, it is proposed to use zero-cost networks (e.g., Bluetooth and WiFi) or relay to data-plan users to upload data.

Mobility and Phone Call Prediction. Many works have been studied on the predictions of human mobility [17] and phone calls [11]. Instead of coming out with new prediction algorithms, effSense leverages these existing approaches to design intelligent uploading schemes to reduce data cost for non-data-plan users and energy consumption for data-plan users.

Problem Statement

Most real-life crowdsensing applications do not need to upload the data immediately after it is sensed. Such applications allow certain delay (a max tolerable delay d_{max}) between collecting data from sensors and uploading data to the server, i.e., the sensed data generated at t_0 on a user's phone can be uploaded during $[t_0, t_0 + d_{max}]$. In this paper, the problem is to design optimal delay-tolerant data uploading schemes that can not only minimize mobile-data-cost for non-data-plan users (U_{ndp}) but also maximally decrease energy-consumption for data-plan users (U_{dp}). We first define two key concepts.

Definition 1 (Critical Events) *A critical event* ($e \in E$) *refers to an encounter that can help non-data-plan users*

[*]3:4 is the ratio when 30 users are selected as data-plan users in MIT Reality Mining, which is the actual project setting [4].

upload data without mobile data cost or/and can help data-plan users save energy on data uploading. E has two subsets:

- *Server-related Critical Events (E_{server}): When one user encounters the server (or intermediate servers, e.g. Bluetooth gateway), the data can be uploaded directly.*
- *User-related Critical Events (E_{user}): When $user_a$ encounters $user_b$ who might be able to better serve data uploading, thus $user_a$ can offload data to $user_b$ for efficacy.*

In each crowdsensing period, users encounter a sequence of critical events ($EVENTS = \{e_1, e_2, \cdots, e_n\}$), where e_1 could be non-data-plan $user_a$ encountering the gateway, e_2 could be non-data-plan $user_b$ encountering data-plan $user_c$. Our effSense framework is designed to provide users with "optimal" decisions at each event (e), either upload data to the server (when $e \in E_{server}$), offload data to the counterpart (when $e \in E_{user}$), or keep the data till next encounter occurs. Now, we formally define the concept of "Event Decision".

Definition 2 (Event Decision) *In delay-tolerant data uploading with maximum delay d_{max}, when a user u_i with local sensed data r_i encounters a critical event e at time t^* ($t^* \in [t_0, t_0 + d_{max}]$), effSense provides a decision whether data r_i needs to be offloaded or kept. We note it $DEC(u_i, r_i, t^*, e, t_0, d_{max}) \rightarrow \{true, false\}$.*

Based on these two definitions, we formulate our main problem as follows:

Problem Formulation: In crowdsensing with max tolerant delay (d_{max}) for data uploading, data plan users (U_{dp}) and non-data plan users (U_{ndp}) encounter a critical event sequence ($EVENTS$). We want to obtain an optimal decision making sequence $DECISIONS$ for $EVENTS$ (i.e., each $d \in DECISIONS$ corresponds to the decision ($true$ or $false$) of an event $e \in EVENTS$), in order to achieve the following two goals dedicated to U_{ndp} and U_{dp}, respectively.

First goal: Maximize the number of non-data-plan users in U_{ndp} whose data is uploaded to the server with zero data cost.

$$\max |\{u | u \in U_{ndp}, R_u(t_0) \in R_{server}(t_0 + d_{max})\}|$$

where $R_u(t_0)$ means the sensed data of user u generated at t_0, and $R_{server}(t + d_{max})$ means the data on the server at $t_0 + d_{max}$.

Second goal: Minimize the energy consumption for all data-plan users during the whole data uploading.

$$\min \sum_{u \in U_{dp}} EnergyConsumed_u(t_0, t_0 + d_{max})$$

where $EnergyConsumed_u(t_0, t_0 + d_{max})$ is the energy consumed in data uploading by the user u during $[t_0, t_0 + d_{max}]$.

It is worth noting that we do not know all events ($EVENTS$) in advance. In real life, the event appears one after another. When an event is encountered, the data offloading decision should be made instantly. Although future events are unknown, they could be predicted to ensure better decision making for data offloading. This will be discussed in more details in the next two sections.

Our Proposed Framework

In order to solve the two-goal optimization problem formulated in the previous section, we design effSense to

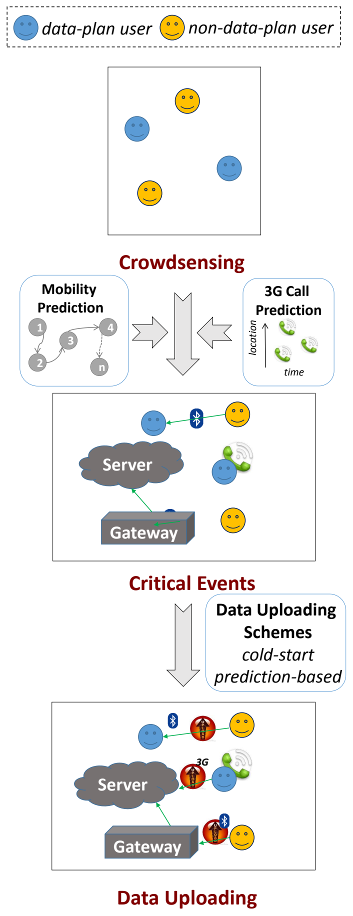


Figure 2: The effSense framework

provide effective uploading schemes during the whole mobile crowdsensing period. Our effSense framework is shown in Figure 2.

In the top part of Figure 2, we observe two types of crowdsensing users, i.e., data-plan users (U_{dp}) and non-data-plan users (U_{ndp}). effSense can identify critical events for each user by using state-of-the-art prediction methods - including *mobility prediction*[17] and *call prediction*[11]. Hereinafter, we obtain a sequence of critical events ($EVENTS$) in the middle of Figure 2. For U_{dp}, critical events can be used to reduce the energy consumption caused by data uploading, e.g., using 3G-call channel. For U_{ndp}, events can be used to eliminate mobile data cost, e.g., meeting Bluetooth gateways and U_{dp}.

In the crowdsensing period, effSense deploys the data uploading schemes for analyzing the critical events. When a user encounters a critical event, effSense applies an optimal scheme to decide whether the user should offload or keep data. As shown in the bottom part of Figure 2, two non-data-plan users send data to a data-plan user and a gateway via Bluetooth respectively, while a data-plan user sends data via 3G when he makes a voice call.

There are two kinds of schemes in effSense for both types of users - *cold-start* and *prediction-based*, which will be explained in the next section.

Note that at the deadline of the data uploading period (i.e. $t_0 + d_{max}$), effSense checks users whether they have non-uploaded data: for U_{ndp} with data, effSense forces them to upload to nearby U_{dp} or gateways if applicable; for U_{dp} with data, effSense forces them to create a new mobile network connection to upload data to the server.

Optimal Uploading Schemes

In this section, we will describe the uploading schemes in detail. Before explaining the specific schemes, we first show some preliminaries.

Preliminaries

Currently, we make the following assumption in effSense:

Assumption - Offload and Dismiss: Once user u offloads his local data, no matter the data receiver is the server, gateway, or another user, u will not be responsible for sending the data any more.

This assumption is meant to avoid redundant data uploading, as one goal of effSense is to reduce energy consumption.

The focus of effSense is to design data uploading schemes by detecting, predicting and exploiting critical events, which are linked to two fundamental definitions, i.e., *event probability* and *event consumption*.

Definition 3 (Event Probability) *Given a user u, a critical event set E^*, a start time t_1, and an end time t_2, the event probability is the possibility that u will encounter any event e ($e \in E^*$) from t_1 to t_2. We represent it as $P_{event}(u, E^*, t_1, t_2)$. Specifically, for the data uploading period with max tolerable deley d_{max}, end time t_2 is fixed ($t_2 = t_0 + d_{max}$), thus we can simplify the notation as $P_{event}(u, E^*, t_1)$.*

Definition 4 (Event Consumption) *Given a user u, local data r, a critical event e, the event consumption is the estimation of the energy that u consumes to upload r under e, represented as $W(u, r, e)$. Specifically, if $e \in E_{user}$, event consumption sums both the energy for sending and receiving data. Sppose that another user is*

u', $W(u,r,e) = W_{send}(u,r,e) + W_{rec}(u',r,e)$.

For the sake of simplification, we do not further discuss the specific methods for predicting event probability and estimating event consumption in this section, as they are not the focus of this paper. The specific methods used in our experiment will be described in the *Evaluation* section.

Uploading Scheme Overview
In general, there are two kinds of schemes for each user type - **cold-start** and **prediction-based**.

- *Cold-start scheme*: It does not require users' history, and applies a straightforward greedy algorithm to recommend users to offload data as soon as he meets a 'friendly' event that can save data cost for non-data-plan users, or save energy consumption for data-plan users.
- *Prediction-based scheme*: It uses users' history to predict future mobility traces and calls to optimize the event decision making.

When a new user joins, effSense can use the cold-start scheme according to his user type to make the event decision. After a period of time, when the user accumulates enough activity logs, effSense can change to the corresponding prediction-based scheme.

Now we begin to explain our proposed schemes in detail. In the following section, suppose that a user u_i encounters a critical event e at time t (another user encountered is u_j if $e \in E_{user}$).

Non-data-plan Uploading Schemes
$\mathbf{SimpleGreedy_{ndp}}$ *(cold-start)*
The intuition behind $SimpleGreedy_{ndp}$ is straightforward.

- If a non-data-plan user encounters an event which can definitely help him upload data to the server without using 3G, then he offloads data.

This kind of events is represented as E_{nd_like}, which includes meeting a data-plan user or a gateway via Bluetooth, connecting to WiFi, etc.

$\mathbf{AdvancedGreedy_{ndp}}$ *(prediction-based)*
The intuition behind $AdvancedGreedy_{ndp}$ is derived from $SimpleGreedy_{ndp}$, while adding a new offloading situation - when non-data-plan u_i meets another non-data-plan u_j who has higher probability to encounter E_{nd_like}, u_i will offload data to u_j. Therefore, u_i will offload data if any of the two following conditions satisfies.

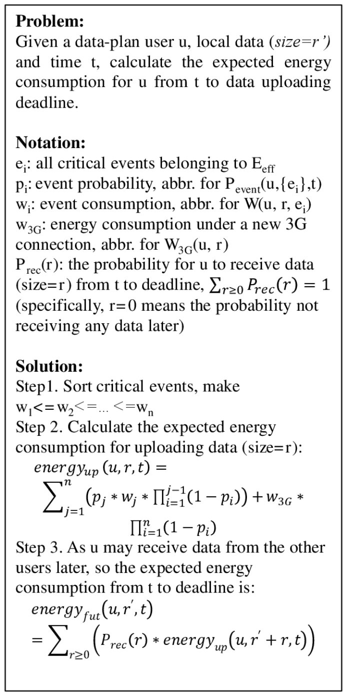
1. $e \in E_{nd\_like}$
2. $e \in \{e' | e' \in E_{user}, u_j \in U_{ndp}\}$ and $P_{event}(u_i, E_{nd\_like}, t) < P_{event}(u_j, E_{nd\_like}, t)$

Data-plan Uploading Schemes
$\mathbf{Greedy_{dp}}$ *(cold-start)*
The intuition behind $Greedy_{dp}$ is making a data-plan user upload data as soon as he meets an event when data could be uploaded to the server via an energy-saving method compared with a new 3G connection. This kind of events is represented as E_{eff}, which is defined as:

$$E_{eff} = \{e | e \in E_{server}, W(u,r,e) < W_{3G}(u,r)\}$$

where $W_{3G}(u,r)$ is the energy consumption for u to upload r by establishing a new 3G connection.

$\mathbf{ExpectationBased_{dp}}$ *(prediction-based)*
The intuition behind $ExpectationBased_{dp}$ can be generally described as follows:



Problem:
Given a data-plan user u, local data (*size=r'*) and time t, calculate the expected energy consumption for u from t to data uploading deadline.

Notation:
e_i: all critical events belonging to E_{eff}
p_i: event probability, abbr. for $P_{event}(u,\{e_i\},t)$
w_i: event consumption, abbr. for $W(u, r, e_i)$
w_{3G}: energy consumption under a new 3G connection, abbr. for $W_{3G}(u, r)$
$P_{rec}(r)$: the probability for u to receive data (size=r) from t to deadline, $\sum_{r\geq 0} P_{rec}(r) = 1$ (specifically, r=0 means the probability not receiving any data later)

Solution:
Step1. Sort critical events, make $w_1 <= w_2 <= \ldots <= w_n$
Step 2. Calculate the expected energy consumption for uploading data (size=r):

$$energy_{up}(u,r,t) = \sum_{j=1}^{n}\left(p_j * w_j * \prod_{i=1}^{j-1}(1-p_i)\right) + w_{3G} * \prod_{i=1}^{n}(1-p_i)$$

Step 3. As u may receive data from the other users later, so the expected energy consumption from t to deadline is:

$$energy_{fut}(u,r',t) = \sum_{r\geq 0}\left(P_{rec}(r) * energy_{up}(u,r'+r,t)\right)$$

Figure 3: Computation for $energy_{fut}$

1. When u_i encounters an event, his expected energy consumption under two conditions are computed - (1) offloading data ($expEnergy_{offload}$), (2) keeping data ($expEnergy_{keep}$).
2. If $expEnergy_{offload} < expEnergy_{keep}$, u_i offloads data; otherwise, u_i keeps data.

Before the detailed explanation, we first give a definition for the expected energy consumption stated previously:

The *expected energy consumption* for a user u at time t is the energy which u is expected to consume from t to the data uploading deadline $t_0 + d_{max}$.

We use the notation $energy_{fut}(u, r', t)$ to represent the expected energy consumption when u keeps data r' locally at time t. The detailed computation process is shown in Figure 3. It is worth noting that in *step 3* , we not only consider the local data r', but also the data that u may receive from other users during $[t, t_0 + d_{max}]$. Therefore, even if u has no local data at t (i.e. $r' = 0$), it is possible that $energy_{fut}(u, 0, t) > 0$.

Now we explain the computation processes of $ExpectationBased_{dp}$. The specific processes are somewhat different between $e \in E_{server}$ and $e \in E_{user}$.

1. $e \in E_{server}$

1) $expEnergy_{offload}$ is divided to two parts - (1) energy for u_i to upload local data r_i, (2) expected energy consumption when u_i has no data at t:

$$expEnergy_{offload} = W(u_i, r_i, e) + energy_{fut}(u_i, 0, t)$$

2) $expEnergy_{keep}$ is exactly the expected energy consumption when u_i keeps data r_i at t:

$$expEnergy_{keep} = energy_{fut}(u_i, r_i, t)$$

2. $e \in E_{user}$

When computing $expEnergy$ here, we take both u_i and u_j into account to get the sum of two users' expected energy consumption.

1) $expEnergy_{offload}$ is extended to two conditions - (1) u_i offloads data to u_j ($expEnergy^{ij}_{offload}$), (2) u_j offloads data to u_i ($expEnergy^{ji}_{offload}$):

$$expEnergy^{ij}_{offload} = W_{send}(u_i, r_i, e) + W_{rec}(u_j, r_i, e) + energy_{fut}(u_i, 0, t) + energy_{fut}(u_j, r_i + r_j, t)$$
$$expEnergy^{ji}_{offload} = W_{send}(u_j, r_j, e) + W_{rec}(u_i, r_j, e) + energy_{fut}(u_j, 0, t) + energy_{fut}(u_i, r_i + r_j, t)$$

2) $expEnergy_{keep}$ means that u_i and u_j both keep data:

$$expEnergy_{keep} = energy_{fut}(u_i, r_i, t) + energy_{fut}(u_j, r_j, t)$$

Through computing $energy_{fut}$ and estimating $W(u, r, e)$, we get the results of these formulas and make the event decision by selecting the condition with the smallest value.

Finally, we describe an implementation issue in effSense.

Info Exchange by Device Name Encoding
In effSense, when two users meet, one user needs to know some info about the other to make the event decision. These info include *user type* (non-data-plan or data-plan), $P_{event}(u, E_{nd\_like}, t)$ (for non-data-plan users), $energy_{fut}$ (for data-plan users), etc. effSense make users exchange the info by encoding them into the device name. One defect is that the device name needs to change occasionally as some info change over time. Fortunately, device name change does not consume much energy.

Table 1: Energy estimation for 3G and Bluetooth (*joule*)

Protocol	Small	Big (xKB)
3G	12	12+0.025x
Bluetooth[‖]	1	1+0.003x

Table 2: Energy estimation for critical events (*joule*)

Event	Small	Big (xKB)
e_{3g_call}[*]	3	3+0.006x
e_{bt_device}	1	1+0.003x
e_{bt_user}[†]	2	2+0.006x

Table 3: N_{nd_upload} on MIT

Scheme	Workday	Holiday	Overall
BT_{act}	26.1	14.8	22.1
SG_{ndp}	23.3 (89%)[‡]	10.6 (71%)	18.8 (85%)
AG_{ndp}	24.3 (93%)	11.0 (74%)	19.6 (89%)

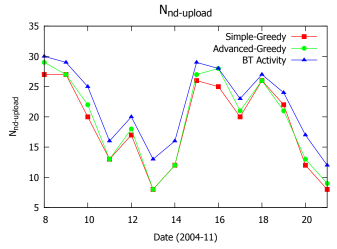


Figure 4: Each day's N_{nd_upload} during 2-week evaluation on MIT

Evaluation

In this section, we evaluate effSense framework using both MIT Reality Mining [5] and Nodobo [2] datasets.

Experimental Setup on MIT

In the MIT dataset, we choose the time period from 2004.10.4 to 2004.11.21 for evaluation, where 71 active users participated in the trial in 7 weeks.. We select top 30 mobile-data-usage users as data-plan users according to the project real setting [4]. We use the first 5 weeks' logs as history, and evaluate effSense on the last 2 weeks. We suppose that the start of each data collection is 00:00:00 and deadline is 23:59:59 each day. So there are 14 continuous data collections during the last 2 weeks.

We choose 3 critical event types - (1) e_{3g_call}: making a 3G call, (2) e_{bt_device}: meeting a Bluetooth gateway [††], (3) e_{bt_user}: meeting another user via Bluetooth.

Based on existing works about the mobile phone energy consumption [1, 10], we construct the energy estimation formulas for both 3G and Bluetooth, as shown in Table 1, for both small-size and big-size data[‡‡]. Further, we use these results to estimate each event energy consumption, as shown in Table 2.

[‖]We do not consider the Bluetooth scan energy consumption, because MIT Reality Mining project itself asks the participants to do frequent Bluetooth scan every 5 minute.

[*]3G data transmission during voice call save at least 75% energy consumption [9].

[†]The energy consumption of e_{bt_user} is twice the Bluetooth energy consumption because of one user sending and one user receiving.

[‡]The values in the brackets are the proportions to the corresponding $N_{nd-upload}$ of BT_{act}.

[††]We choose two devices named *localhost.media.mit.edu* and *studies.media.mit.edu* as gateways.

[‡‡]Small-size data uploading consumes almost the same energy no matter what is the specific data size[1].

Prediction Method

Our experiment applied a simple timeslot-based prediction method. We split one week into $24 * 7$ timeslots so that each timeslot lasts one hour. The calculation process for $P_{event}(u, E, t_1, t_2)$ is:

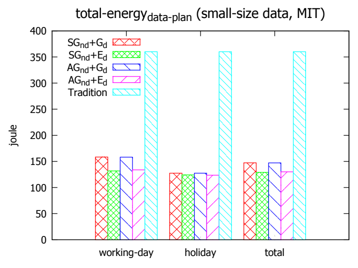
1. Map t_1, t_2 to the corresponding timeslots ts_1 and ts_2.
2. Suppose that the history data includes m weeks, and user u encounters any event $e \in E$ from ts_1 to ts_2 in n weeks, then $P_{event}(u, E, t_1, t_2) = n/m$

Experiment Results on MIT

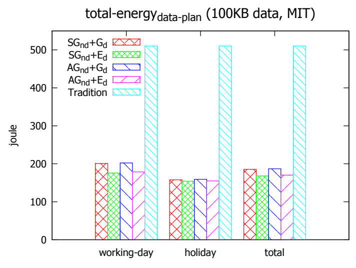
Corresponding to the two optimization goals of effSense, we primarily want to answer two questions - (1) How many non-data-plan (NDP) users can upload data without data cost? (2) How much energy can data-plan (DP) users save in data uploading in effSense? The answers for these two questions are also directly related to the performances of our proposed NDP and DP uploading schemes, respectively.

Performance of Data Cost Conservation on MIT

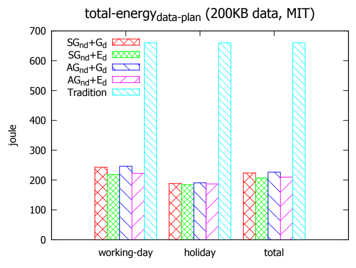
First, we investigate the number of NDP users who upload data successfully (N_{nd_upload}) for two NDP schemes (Figure 4 and Table 3). In general, effSense can help 45%∼48% NDP users upload data successfully without data cost (i.e. help average 18.8∼19.6 in total 41 NDP users in one data collection). In Figure 4, effSense did not perform well in the holidays (day 11, 13, 14, 20, and 21 in Nov. 2004), as few users came to school so that the opportunities for Bluetooth relay dropped. In Table 3, we introduce an upper bound of N_{nd_upload}, which is the number of NDP users having Bluetooth activity (BT_{act}), because only this kind of NDP users might upload data via Bluetooth relay to cut data cost. Further, $AdvancedGreedy_{ndp}$ outperformed about 4% over $SimpleGreedy_{ndp}$. As $SimpleGreedy_{ndp}$ achieved nearly



(a) small



(b) 100KB



(c) 200KB

Figure 5: $totalEnergy_{dp}$ for small/100KB/200KB data on MIT

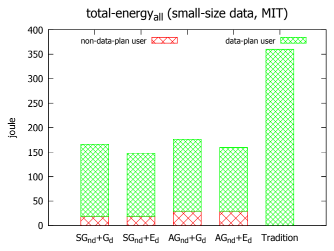


Figure 6: $totalEnergy_{all}$ for small-size data on MIT

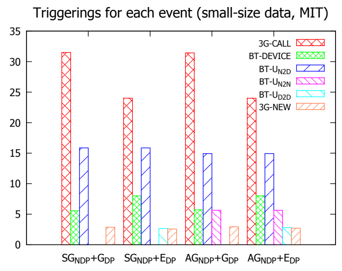


Figure 7: Event triggerings in one data collection on MIT

90% performance compared with the upper bound BT_{act} on workdays, this improvement was still meaningful.

Performance of Energy Conservation on MIT

Figure 5 shows the energy consumption for DP users in data uploading ($totalEnergy_{dp}$) of three data sizes (small, 100KB, and 200KB). effSense saved 55%∼65% of the energy consumption for DP users compared with the traditional method[‡]. Specifically, $ExpectationBased_{dp}$ can save 9.8%∼17.5% more energy compared with $Greedy_{dp}$ on workdays (given the same NDP scheme).In addition, as the data size increases, the performance gap between $ExpectationBased_{dp}$ and $Greedy_{dp}$ decreases. The reason is that as the data size increases, the overhead for one user to help others relay data becomes larger.

Figure 6 shows the total energy consumption for both DP and NDP users ($totalEnergy_{all}$). Though effSense incurs NDP users' energy consumption which does not exist in the traditional method, $totalEnergy_{all}$ is still reduced by 51%∼56%.

Event Triggering Times on MIT

Figure 7 shows the actual triggering times to offload data for each kind of events. Specifically, e_{user} is divided into three senarios according to the data offloading direction: (1) $U_{ndp} \rightarrow U_{dp}$ (BT_U_{N2D}), (2) $U_{ndp} \rightarrow U_{ndp}$ (BT_U_{N2N}), and (3) $U_{dp} \rightarrow U_{dp}$ (BT_U_{D2D}).[*]

First, the triggering times of establishing a new 3G connection decreased significantly in effSense compared with the traditional method which would incur 30 new 3G connections in every data collection. As establishing a new 3G connection is energy-demanding, this is the primary reason why effSense can save energy. In addition, for the cold-start scheme pair $\{SimpleGreedy_{ndp}, Greedy_{dp}\}$, the data exchanges between users are only carried out as $U_{ndp} \rightarrow U_{dp}$. However, $AdvancedGreedy_{ndp}$ and $ExpectationBased_{dp}$ introduced the data exchanges of

[‡]The traditional method means that data-plan users upload their data via a new 3G connection everyday, while non-data-plan users store their data in the SD-cards instead of uploading to the server.

[*]$U_{dp} \rightarrow U_{ndp}$ does not exist in effSense because this direction completely conflicts the goal of saving data cost for U_{ndp}

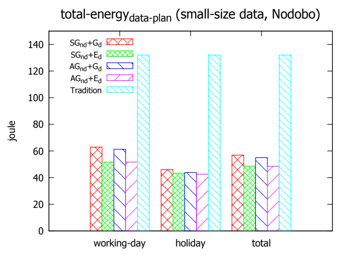


Figure 8: $totalEnergy_{dp}$ for small-size data on Nodobo

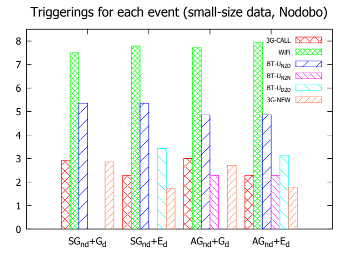


Figure 9: Event triggerings in one data collection on Nodobo

Table 5: N_{nd_upload} on Nodobo

Scheme	Workday	Holiday	Overall
BT_W_{act}	10.33	7.20	9.21
SG_{ndp}	10.00 (97%)	6.20 (86%)	8.64 (94%)
AG_{ndp}	10.33 (100%)	6.40 (89%)	8.93 (97%)

$U_{ndp} \rightarrow U_{ndp}$ and $U_{dp} \rightarrow U_{dp}$, respectively. Furthermore, given the same NDP scheme, $ExpectationBased_{dp}$ triggers much fewer e_{3g_call} and more e_{bt_device} than $Greedy_{dp}$. This is why $ExpectationBased_{dp}$ can save more energy than $Greedy_{dp}$, as energy consumption for e_{bt_device} is less than e_{3g_call}.

Results on Other Delay Settings

Besides the 24-hour delay assumed, we conduct some experiments with shorter delays* (Table 4). With the delay decreasing, NDP users have fewer chances to upload data without data cost (N_{nd_upload} decreases) and DP users consume more energy ($totalEnergy_{dp}$ increases), because few critical events, which effSense can use to improve users' experience, exist in a short delay. In addition, for a short delay, collection start time affects N_{nd_upload} and $totalEnergy_{dp}$, as the number of critical events may differ greatly between different time periods.

Table 4: N_{nd_upload} and $totalEnergy_{dp}$ for different delays

Delay	N_{nd_upload}	$totalEnergy_{dp}$
3-hour	10.7	178.9J
6-hour	14.7	167.0J
12-hour	18.0	140.3J
24-hour	19.6	130.4J

Evaluation on Nodobo

We also evaluate effSense on Nodobo, as this dataset contains users' WiFi traces, which MIT Reality Mining lacks. Therefore, we introduce e_{wifi}, which means a user connecting to a WiFi AP. e_{wifi} can help NDP users upload data without data cost. The energy consumption to upload small-size data via WiFi was set to $2J$ [1]. We exclude e_{bt_device} as no gateways exist in Nodobo. The other experimental settings are small-size data, 24-hour delay, 11 DP users and 16 NDP users.

*Experimental settings: small data size and the prediction-based schemes $\{AdvancedGreedy_{ndp}, ExpectationBased_{dp}\}$.

The results are shown in Figure 8, Figure 9, and Table 5. In Table 5, we set the upper bound of N_{nd_upload} as the number of NDP users who have either Bluetooth or WiFi activity, called BT_W_{act}. The results are similar to MIT Reality Mining. Figure 9 shows that the new event e_{wifi} was effective in improving the overall performance of effSense, as e_{wifi} was triggered most frequently. This reveals the potential that effSense can become more powerful when introducing more useful critical events.

Discussion

Ratio of data-plan and non-data-plan users

The ratio of data-plan and non-data-plan users will affect the performance of effSense. Currently we choose 30 users as data-plan users, the same as actual MIT Reality Mining project setting[4] to simulate the real-life condition. Though the thorough experiments under various ratios are not covered in this paper, the proposed mechanisms are still valuable for real-life mobile crowdsensing tasks.

Data Size

In the experiment, we set three data sizes - small, 100KB, and 200KB. In fact, some sensing tasks can generate large data up to several MB (e.g. photos). When considering such large data, we will face some additional problems. For instance, we cannot simply make the assumption that two users can succeed in exchanging several MB data via Bluetooth when they meet, because the opportunistic Bluetooth encounter is temporary and unstable. In this paper, we did not conduct the experiment for large data up to MB level. effSense should be improved in our future work to handle large data collection issue.

Energy Consumption vs Battery Life

For current smartphones, uploading small-size data via 3G once only consumes about 0.1% battery (e.g. Nokia N95 with a 950mAh battery). However, energy-saving schemes still play a role in effSense, because the battery drain can increase for two reasons. First, more uploadings (i.e. shorter delay) and larger data will drain more battery. Second, the most 'friendly' data-plan users will help other 2∼3 non-data-plan users upload data per collection according to our experiment, which incurs extra battery drain. E.g., as six 200KB data uploadings in the daytime (i.e. about 3-hour delay) drain about 1.5% battery, the most 'friendly' data-plan users will drain up to 5% battery per day in data uploading. By introducing energy-saving schemes into effSense, the most 'friendly' data-plan users can decrease the battery drain to around 1%.

Additional Critical Events

Due to the dataset limitation, we cannot capture all the useful critical events in the experiment. For example, whether a phone is charging or not plays a vital role in deciding if data should be offloaded or kept. In our future work, we plan to introduce the phone charging event and other useful events into effSense.

Other Issues to Improve effSense

We can also improve effSense in many other directions, such as using a more precise energy estimation mechanism and an advanced user mobility/call prediction method.

Conclusion

In this paper, we investigate how to reduce both energy consumption (for data-plan users) and mobile data cost (for non-data-plan users) caused by data uploading in mobile crowdsensing, and design the *effSense* framework to improve both users' experience in delay-tolerant mobile crowdsensing. The experiment results on MIT Reality and Nodobo datasets verified the effectiveness of effSense.

References

[1] Balasubramanian, N., Balasubramanian, A., and Venkataramani, A. Energy consumption in mobile phones: a measurement study and implications for network applications. In *IMC* (2009), 280–293.

[2] Bell, S., McDiarmid, A., and Irvine, J. Nodobo: Mobile phone as a software sensor for social network research. In *Vehicular Technology Conference* (2011), 1–5.

[3] Chu, D., Lane, N. D., Lai, T. T.-T., Pang, C., Meng, X., Guo, Q., Li, F., and Zhao, F. Balancing energy, latency and accuracy for mobile sensor data classification. In *SenSys* (2011), 54–67.

[4] Eagle, N. The reality mining data readme: http://www.media.mit.edu/ventures/EPROM/data/RealityMining_ReadMe.pdf.

[5] Eagle, N., and (Sandy) Pentland, A. Reality mining: sensing complex social systems. *Personal Ubiquitous Comput. 10* (2006), 255–268.

[6] Hull, B., Bychkovsky, V., Zhang, Y., Chen, K., Goraczko, M., Miu, A., Shih, E., Balakrishnan, H., and Madden, S. CarTel: a distributed mobile sensor computing system. In *SenSys* (2006), 125138.

[7] Kjrgaard, M. B., Bhattacharya, S., Blunck, H., and Nurmi, P. Energy-efficient trajectory tracking for mobile devices. In *MobiSys* (2011), 307–320.

[8] Musolesi, M., Piraccini, M., Fodor, K., Corradi, A., and Campbell, A. T. Supporting energy-efficient uploading strategies for continuous sensing applications on mobile phones. In *Pervasive Computing*, P. Floren, A. Krger, and M. Spasojevic, Eds., no. 6030 in LNCS. 2010, 355–372.

[9] Nurminen, J. Parallel connections and their effect on the battery consumption of a mobile phone. In *CCNC* (2010), 1–5.

[10] Perrucci, G., Fitzek, F., and Widmer, J. Survey on energy consumption entities on the smartphone platform (2011). 1–6.

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

[12] Ra, M.-R., Paek, J., Sharma, A. B., Govindan, R., Krieger, M. H., and Neely, M. J. Energy-delay tradeoffs in smartphone applications. In *MobiSys* (2010), 255–270.

[13] Rachuri, K. K., Mascolo, C., Musolesi, M., and Rentfrow, P. J. SociableSense: exploring the trade-offs of adaptive sampling and computation offloading for social sensing. In *MobiCom* (2011), 73–84.

[14] Rana, R. K., Chou, C. T., Kanhere, S. S., Bulusu, N., and Hu, W. Ear-phone: an end-to-end participatory urban noise mapping system. In *IPSN* (2010), 105–116.

[15] Sheng, X., Tang, J., and Zhang, W. Energy-efficient collaborative sensing with mobile phones. In *INFOCOM* (2012), 1916–1924.

[16] Sherchan, W., Jayaraman, P. P., Krishnaswamy, S., Zaslavsky, A., Loke, S., and Sinha, A. Using on-the-move mining for mobile crowdsensing. In *MDM* (2012), 115–124.

[17] Vu, L., Do, Q., and Nahrstedt, K. Jyotish: A novel framework for constructing predictive model of people movement from joint wifi/bluetooth trace. In *PerCom* (2011), 54–62.