effSense: A Novel Mobile Crowd-Sensing Framework for Energy-Efficient and Cost-Effective Data Uploading

Leye Wang, Daqing Zhang, *Member, IEEE*, Zhixian Yan, *Member, IEEE*, Haoyi Xiong, *Member, IEEE*, and Bing Xie

***Abstract*—Energy consumption and mobile data cost are two key factors affecting users' willingness to participate in mobile crowd-sensing tasks. While data-plan (DP) users are mostly concerned with energy consumption, non-data-plan (NDP) users are more sensitive to data cost. Traditional ways of data uploading in mobile crowdsensing tasks often go to two extremes: either in real time or completely offline after the whole task is over. In this paper, we propose effSense—an energy-efficient and cost-effective data uploading framework, which utilizes adaptive uploading schemes within fixed data uploading cycles. In each cycle, effSense empowers the participants with a distributed decision making scheme to choose the appropriate timing and network to upload data. effSense reduces data cost for NDP users by maximally offloading data to Bluetooth/WiFi gateways or DP users encountered; it reduces energy consumption for DP users by piggybacking data on a call or using more energy-efficient networks rather than initiating new 3G connections. By leveraging the predictability of users' calls and mobility, effSense selects proper uploading strategies for both user types. Our evaluation with the MIT reality mining and Nodobo datasets shows that effSense can reduce 55%–65% energy consumption for DP users, and 48%–52% data cost for NDP users, respectively, compared to traditional uploading schemes.**

***Index Terms*—Data uploading, delay-tolerant crowd sensing, energy saving, mobile crowdsensing, mobile data usage.**

I. Introduction

As SENSOR-equipped smartphones get more and more popular, mobile crowd sensing [1], [2] has become an effective way to carry out various sensing tasks such as environmental monitoring [3] and social sensing [4]. To encourage users to participate in mobile crowdsensing tasks, it is paramount to minimize the inconvenience incurred for users. In this regard, energy consumption and mobile data cost are two critical concerns. While energy consumption is related to a mobile phone's battery life, mobile data cost is associated with the fees incurred, especially for the users who do not hold an unlimited data plan. Therefore, reducing energy consumption and data cost incurred can encourage more people to actively participate in crowdsensing tasks.

Researchers have developed several energy-saving approaches for attracting engagements in mobile crowdsensing. The proposed solutions include adopting dynamic sensing duty cycle [4], making a trade-off between local and remote computation [5], reducing data uploading frequency by predicting missing data on the server side [6], and splitting the task intelligently among users [7], etc.

These existing works mostly assume that the sensed data should be sent to a central server as soon as the data is produced. In fact, some mobile crowdsensing tasks do not necessarily require the sensed data to be uploaded in real time. For example, in the MIT reality mining project [8], around 100 participants' mobile traces were collected to understand users' interests, activity patterns, etc. The project collected users' data in two ways.

1) Thirty participants were provided with a mobile data plan [9]. These users uploaded their sensed data every night.
2) The other participants' data was stored on SD-cards and was collected after the mobile phones were returned at the end of the project.

For both types of participants, a certain amount of time delay between sensing and uploading is allowed.

In this paper, for the crowdsensing task that does not require real-time sensed data uploading (called a delay-tolerant crowdsensing task), we 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 a data plan or who are not willing to use their data plan for crowdsensing tasks (called non-data-plan (NDP) users) to reduce data cost by relaying data to a Bluetooth gateway or other mobile phones encountered (rather than via 3G network) and 2) users with a data plan (called DP users) to consume less energy in data uploading.

With the mobile users classified into two groups with different optimization goals: NDP users (reducing data cost) versus DP users (reducing energy consumption), we propose

Manuscript received September 22, 2014; revised December 21, 2014; accepted January 19, 2015. Date of publication May 19, 2015; date of current version November 13, 2015. This work was supported by the National High Technology Research and Development Program of China (863) under Grant 2013AA01A605. This paper was recommended by Associate Editor J. Lu.

L. Wang, D. Zhang, and H. Xiong are with the Department of Telecommunication Network and Services, Institut Mines-TELECOM/TELECOM SudParis, Évry 91011, France (e-mail: daqing.zhang@telecom-sudparis.eu).

Z. Yan is with Go Daddy, Sunnyvale, CA 94089 USA.

B. Xie is with Peking University, Beijing 100871, China.

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

Digital Object Identifier 10.1109/TSMC.2015.2418283

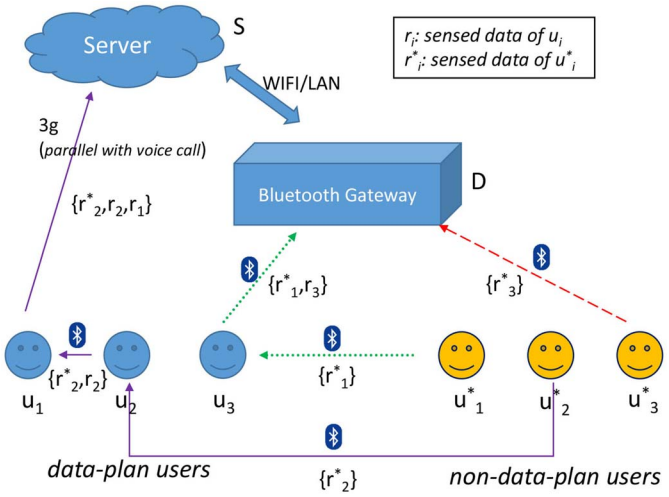


Fig. 1. Running example of effSense.

to change the data uploading scheme in effSense from real time to allowing a certain amount of delay with fixed uploading cycles. In such a way, sensed data uploading in mobile crowdsensing tasks becomes delay-tolerant and data only needs to be sent to the central server before the end of each data uploading cycle (rather than immediately after it is produced). As effSense allows a certain amount of time delay between data sensing and data uploading, mobile participants might encounter each other or cheaper networks/devices to relay the sensed data before the end of each uploading cycle, so that their data cost or energy consumption can be preserved. Specifically, effSense empowers each mobile device with a distributed data relaying/uploading scheme to decide when and how to upload the sensed data, in order to reduce data cost for NDP users and energy consumption for DP users.

effSense is designed based on the following observations.

1) NDP users can eliminate mobile data cost in data uploading by using zero-cost networks 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) DP users can reduce energy consumption in data uploading via the energy-efficient methods other than establishing a new 3G connection. For example, piggybacking a data uploading task on a 3G voice call can save 75%–90% energy consumption [10]. Alternatively, uploading data via WiFi or Bluetooth consumes less energy than via normal 3G.

A. Running Example

Fig. 1 shows a simple example to illustrate the basic idea of effSense. Here, u_1, u_2, and u_3 are three DP users, respectively, while u_1^*, u_2^*, and u_3^* are three NDP users, respectively. 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 offers the following data uploading paths to save data cost and energy consumption.

1) *Path 1 (Red Dashed Line):* $u_3^* \rightarrow D \rightarrow S$. A NDP user ($u_3^*$) uploads data via a Bluetooth gateway.
2) *Path 2 (Green Dot Line):* $u_1^* \rightarrow u_3 \rightarrow D \rightarrow S$. A NDP user ($u_1^*$) relays data to a DP user (u_3) first and then u_3 uploads data via a Bluetooth gateway other than 3G directly to reduce both data cost and energy consumption.
3) *Path 3 (Purple Solid Line):* $u_2^* \rightarrow u_2 \rightarrow u_1 \rightarrow S$. The primary difference between this path and the two above is that a DP user (u_1) piggybacks the data uploading task over a voice call in the end.

As shown in this example, NDP users are guided to upload their sensed data without mobile data cost, while DP users are recommended to upload data via energy-efficient methods. Such preservation of data cost and energy consumption is exactly the design objective of effSense.

The key issues involved in designing effSense include the following.

1) Identify and predict critical events. Typical critical events include making a voice call, meeting another user, encountering a Bluetooth gateway, connecting to a WiFi access point (AP), etc. Predicting the critical events for each user is the basis for effSense to select the right data relaying strategy. By leveraging the state-of-the-art activity prediction methods [11], [12], effSense can predict critical events accurately.
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. For instance, uploading a data packet smaller than 10 kB via 3G always consumes about 12 J, whatever the exact data size [13]. According to the existing literature about the energy consumption of mobile phones [13], [14], we estimate the energy consumption of various critical events for different data sizes.
3) Design real-time algorithms to decide if the 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 consumption or data cost.

In summary, this paper has the following contributions.

1) To the best of our knowledge, this is the first work that aims to minimize both energy consumption and mobile data cost in mobile crowdsensing tasks by leveraging heterogeneous networks and delay-tolerant mechanisms.
2) We consider two types of users (NDP and DP) with different goals and thus propose different data uploading schemes for each kind of users: one is purely greedy, the other is based on the mobility/call predictions. While the former one is quite effective in handling the cold-start problem when the participants' historic call or mobility logs are not available, the latter one can achieve better performance by leveraging critical-event prediction according to the participants' historic logs.
3) We evaluate effSense with two real-world datasets—MIT reality mining [8] and Nodobo [15].

The results show that effSense could upload about 48%–52% of NDP users' data without extra data cost, and reduce 55%–65% of DP users' data-uploading energy consumption compared to the traditional method, given the condition that DP users and NDP users have the ratio of 3:4[1] and a data uploading cycle of 24 h.

II. Related Work

A. Mobile Crowd-Sensing Framework

Recent studies on mobile crowd sensing lead to many applications [1], [2], such as urban noise monitoring [3] and social interaction sensing [4]. To enable these applications, the crowdsensing frameworks covering participatory management [16], [17], data uploading strategies [5], [6], programming interface [18], etc., have been studied. This paper also provides a mobile crowdsensing framework, but the research objective is different from the existing work as we address both NDP and DP users' cost and energy concerns in data uploading. effSense could also be integrated into state-of-the-art mobile crowdsensing platforms, such as ParticipAct [19], to improve the user experience.

B. Energy Conservation in Mobile Crowdsensing

Designing an intelligent task assignment algorithm to minimize the number of sensing participants can reduce the total energy consumption in a mobile crowdsensing task. For instance, to select the minimal number of participants to cover a restricted area, many mechanisms are proposed based on different task models [7], [16], [17], [20], [21].

For each individual device in mobile crowdsensing, energy saving mechanisms have been researched in three phases: 1) sensing; 2) computing; and 3) data uploading.

The first two phases, sensing and computing, are responsible for the process of acquiring and processing data from sensors. To reduce the energy consumption of sensing, existing approaches include adopting low power sensors [22], adaptive sensor schedulers [23], [24], etc. To reduce the energy consumption of computing, mobile sensing systems can use low power processors [25], code offloading [26], etc. While effSense focuses on the uploading phase which is complementary to these two phases, all the aforementioned approaches can be incorporated into effSense to make crowdsensing more energy-efficient.

To reduce the energy consumption in data uploading, the following approaches are proposed by previous work.

1) Use low power communication methods [27]–[29]. effSense follows this in leveraging Bluetooth, WiFi, or parallel data uploading with voice calls [10] as energy-efficient communication methods, rather than directly using 3G.
2) Use mobile nodes [30] to carry and forward data (relays between sensing devices) to the server. effSense engages "active" DP users (i.e., users who have better chance to upload data via energy-efficient methods) as relays to help "inactive" DP users, to reduce total energy consumption.
3) Upload less data, e.g., compressing data before uploading [31], or uploading part of the data while deducing the rest [6]. These approaches consume extra energy in computation, so the decision to apply these techniques needs to be studied carefully to see the overall performance. Currently, effSense does not apply these mechanisms.

[1]3:4 is the ratio when 30 users are selected as DP users in the MIT reality mining project, which is the actual project setting [9].

C. Data Cost Conservation in Data Uploading

To decrease data cost in uploading, previous work focuses mainly on reducing data size via additional computation on the local mobile devices to aggregate sensed data before uploading [4], [5]. Moreover, the energy-saving work such as [6] and [31] also aims at reducing the amount of the data to upload. While these existing approaches can reduce data cost significantly when the data size is big, they cannot eliminate it completely. As effSense intends to eliminate data cost for NDP users, we thus propose to use zero-cost networks (e.g., Bluetooth and WiFi) or relay to DP users to upload data.

D. Mobility and Phone Call Prediction

Much work has been done on the prediction of human mobility [11], [32] and phone calls [12], [33]. Instead of coming out with new prediction algorithms, effSense leverages existing approaches based on the Poisson distribution [11], [12] to design intelligent uploading schemes to reduce data cost for NDP users and energy consumption for DP users.

III. Problem Statement

Many real-life crowdsensing applications (e.g., MIT reality mining [8] and environment monitoring [20], [21]) do not need to upload the sensed data immediately after it is sensed and collected. Such applications allow some delay (a max tolerable amount of delay $d_{\max}$) between collecting the data from sensors and uploading it to the server, i.e., the sensed data generated at t_0 on a user's phone can be uploaded during $[t_0, t_d]$ (where $t_d = t_0 + d_{\max}$). In this paper, the research goal is to design the delay-tolerant data uploading schemes that can not only minimize mobile data cost for NDP users (U_{ndp}) but also maximally decrease energy consumption for DP users (U_{dp}).

For this paper, we make the following assumptions in the crowdsensing process.

Assumption 1 (Offload and Dismiss): Once a user u offloads the sensed data to a recipient, no matter the recipient is the server, gateway, or another user, u will not be responsible for sending the data any more.

This assumption in effSense ensures having only one copy for all the sensed data and thus avoids redundant data uploading to meet the energy-saving objective.

Assumption 2 (Offload All Data): Once a user catches a chance to offload data, all the data will be offloaded to the recipient, no matter the data was sensed locally or received from other users.

This assumption seems quite strong at first glance, especially when we use very short Bluetooth encounters to transfer a large amount of data. However, there exist many crowdsensing applications that generate small amount of data; for such applications, this assumption holds most of the time. For example, after analyzing the user data collected in the MIT reality mining project, we find that even using the plain text to store users' sensed data without compression, the amount of data for one user per day is less than 100 kB. Section VII-A will discuss this issue in more details.

Before formulating the problem formally, we introduce some key concepts.

Definition 1 (Critical Events): A critical event ($e \in E$) refers to an encounter between a NDP mobile terminal and another device that can help NDP users offload data without cost, or/and a call/encounter that can help DP users save energy consumption in data uploading. The critical event set E contains two subsets.

1) *Server-Related Critical Events (E_s):* When a user encounters a server (including intermediate servers, e.g., Bluetooth gateway) or initiates/receives a call, the sensed data can be uploaded to the server directly.
2) *User-Related Critical Events (E_u):* When user_a encounters user_b who might be able to better accomplish data uploading, user_a can offload data to user_b to reduce data cost or energy consumption.

In each crowdsensing cycle, users encounter a sequence of critical events (EVENTS $= \{e_1, e_2, \ldots, e_n\}$), where e_1 could be user_a encountering a server, e_2 could be NDP user_b encountering DP user_c, and e_n could be DP user_d receiving a phone call. Our effSense framework is designed to provide users with decisions at each event (e), either upload the data to the server (when $e \in E_s$) or offload the data to an encountered mobile device (when $e \in E_u$), or keep the data till next critical event occurs. Now, we formally define the "decision making" mechanism.

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

As the mobile data cost and energy consumption are two primary concerns when critical events occur, we define "event data cost function" and "event energy consumption" as follows.

Definition 3 (Event Data Cost Function): The event data cost function represents whether an event e incurs mobile data cost or not

$$\delta(e) = \begin{cases} \text{true}, & \text{if } e \text{ incurs data cost} \\ \text{false}, & \text{otherwise.} \end{cases}$$

Definition 4 (Event Energy Consumption): 1) for $e \in E_s$, the event energy consumption is the amount of energy that user u consumes to upload r to the server under e, marked as $W_s(u, r, e)$ and 2) for $e \in E_u$, the total event energy consumption comprises two parts—user u_i sending data and user u_j receiving data, marked as $W_u(u_i, u_j, r, e) = W_{u\_sd}(u_i, r, e) + W_{u\_rv}(u_j, r, e)$.

As critical event prediction is helpful in event decision making, we also define "event probability" as follows.

Definition 5 (Event Probability): Given a user u, a critical event set E^*, a time t, the event probability is the probability that u will encounter any event e ($e \in E^*$) from t to the end of data uploading cycle t_d (i.e., $t_0 + d_{\max}$), marked as $P_e(u, E^*, t, t_d)$.

Based on these definitions, we formulate our problem as follows.

Problem Statement: In a crowdsensing task with some delayed uploading cycle (max delay $d_{\max}$), DP users (U_{dp}) and NDP users (U_{ndp}) would encounter a critical event sequence (EVENTS). Each mobile device aims to obtain a decision making sequence DECISIONS corresponding to EVENTS [i.e., each $d \in$ DECISIONS corresponds to the decision (true or false) at an event $e \in$ EVENTS], in order to achieve the following two goals dedicated to U_{ndp} and U_{dp}, respectively.

1) *First Goal:* Maximize the number of NDP users whose data is uploaded to the server with zero data cost

$$\max\left|\left\{u | u \in U_{\text{ndp}}, R_u(t_0) \in R_s(t_d)\right\}\right|$$

where:
$R_u(t_0)$ sensed data of u produced at t_0;
$R_s(t_d)$ sensed data on the server at t_d.

2) *Second Goal:* Minimize the energy consumption for DP users during the data uploading process

$$\min \sum_{u \in U_{\text{dp}}} \text{EnergyCons}_u(t_0, t_d)$$

where $\text{EnergyCons}_u(t_0, t_d)$ is the amount of energy that u consumes in data uploading during $[t_0, t_d]$.

It is worth noting that we do not know when the critical events (EVENTS) would occur in advance. In practice, the event appears one after another. When an event occurs for a mobile device, a decision should be made instantly in a distributed manner. Although future events are unknown, they could be predicted to help decision making for data offloading.

IV. effSense Framework

In order to solve the two-goal optimization problem formulated in the previous section, we design effSense to accomplish effective data uploading for mobile crowdsensing applications. Our effSense framework is shown in Fig. 2.

As shown in the left part of Fig. 2, we have two types of crowdsensing users, i.e., DP (U_{dp}) and NDP users (U_{ndp}). Each mobile device identifies future critical events in a distributed manner using state-of-the-art prediction techniques—for both mobility prediction [11] and call prediction [12]. Accordingly, we obtain a sequence of critical events (EVENTS) for each mobile device in the middle of Fig. 2. For U_{dp}, critical events are used to reduce energy consumption by offloading data via Bluetooth gateways encountered or piggybacking sensed data on a 3G phone call as predicted. For U_{ndp}, critical events are used to eliminate mobile data cost by offloading data to Bluetooth gateways or U_{dp} devices encountered as predicted.

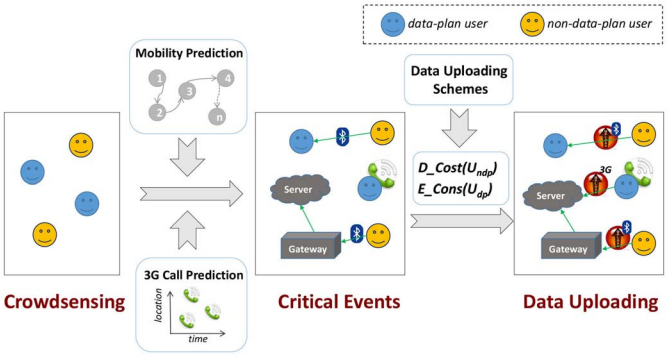


Fig. 2. effSense framework.

In each crowdsensing uploading cycle, effSense selects the data uploading schemes by analyzing the critical events. When a user encounters a critical event, effSense makes the decision to offload or keep the data. As shown in the right part of Fig. 2, the generated data uploading schemes might include two NDP users sending data to a DP user and a gateway via Bluetooth, respectively, and a DP user sending data via 3G when he makes a voice call.

There are two kinds of schemes proposed in effSense for both types of users.

1) *Cold-Start Scheme:* It does not require users' historic event traces, and applies a straightforward greedy algorithm to offload data as soon as it encounters a "promising" event that can eliminate data cost for NDP users $D\_\text{Cost}(U_\text{ndp})$, or reduce energy consumption for DP users $E\_Cons(U_\text{dp})$.
2) *Prediction-Based Scheme:* It compares the current uploading cost (i.e., offload data at current event) with future predicted uploading cost (i.e., keep data at current event), to decide whether the sensed data should be offloaded or kept.

Before the end of each data uploading cycle, effSense checks with all mobile devices to see whether they have unuploaded data: for U_ndp with data, effSense forces them to offload data to nearby U_dp if possible; for U_dp with data, effSense forces them to create a 3G connection to upload data.

V. Uploading Schemes

We propose two uploading schemes (cold-start and prediction-based) for both NDP and DP users. Note that the event probability (P_e) prediction is not the focus of this paper, so the prediction method will be later described in the experiment (Section VI). The important notations are listed in Table I.

A. Uploading Schemes for Non-Data-Plan Users

Suppose a NDP user u_i encounters a critical event e at time t (the encountered user is u_j if e is user-related), we propose two schemes for NDP users to offload data: 1) $\text{SimpleGreedy}_\text{ndp}$ (cold-start) and 2) $\text{AdvancedGreedy}_\text{ndp}$ (prediction-based).

1) $\text{SimpleGreedy}_{ndp}$: $\text{SimpleGreedy}_\text{ndp}$ follows the logic below.

a) When a NDP user u_i encounters a server-related event:
   i) if the event is an encounter with a Bluetooth gateway or a WiFi AP, u_i uploads the sensed data as it will not incur any data cost;
   ii) if the event is a 3G call, u_i will not upload data, as piggybacking data on a call only reduces energy but still incurs 3G data cost.
b) When a NDP user u_i encounters a user-related event:
   i) if the encountered user u_j is a DP user, u_i offloads data to u_j, because DP users can ensure uploading data before the uploading cycle deadline;
   ii) if the encountered user u_j is a NDP user, u_i does not offload data.

For generality, we use $E_\text{ndp\_like}$ to represent the above events that make the decision making true

$$E_\text{ndp\_like} = \{e | e \in E_s, \delta(e) = \text{ false}\} \cup \{e | e \in E_u, \delta(e) = \text{false}, u_j \in U_\text{dp}\}.$$

2) $\text{AdvancedGreedy}_{ndp}$: On top of $\text{SimpleGreedy}_\text{ndp}$, $\text{AdvancedGreedy}_\text{ndp}$ adds a new data offloading condition when a NDP user u_i meets another NDP user u_j.

a) When a NDP user u_i meets another NDP user u_j, if u_j has higher probability to meet DP users or to upload data via Bluetooth or WiFi gateways (i.e., encountering $e \in E_\text{ndp\_like}$) than u_i, u_i will offload data to u_j.

So, the events which would trigger data offloading are generalized as follows.

a) $e \in E_\text{ndp\_like}$ (same as $\text{SimpleGreedy}_\text{ndp}$).
b) $e \in \{e' | e' \in E_u, u_j \in U_\text{ndp}\}$ and $P_e(u_i, E_\text{ndp\_like}, t, t_d) < P_e(u_j, E_\text{ndp\_like}, t, t_d)$ (new'data offloading condition).

B. Uploading Schemes for Data-Plan Users

Suppose that a DP user u_i encounters a critical event e at time t (the encountered user is u_j if e is user-related), we design two energy-saving schemes for

TABLE I
Notations

symbol	*definition*
u	a user
r	sensed data size
e	a critical event
t_0	sensed data generated time (i.e., uploading cycle start time)
d_{max}	max tolerable delay
t_d	uploading cycle deadline (i.e., $t_0 + d_{max}$)
U_{ndp}	non-data-plan users
U_{dp}	data-plan users
E_s	server-related critical events
E_u	user-related critical events
$\delta(e)$	whether e incurs data cost or not
$W_s(u, r, e)$	energy consumption for u to upload r when $e \in E_s$
$W_{u\_sd}(u, r, e)$	energy consumption for u to send r when $e \in E_u$
$W_{u\_rv}(u, r, e)$	energy consumption for u to receive r when $e \in E_u$
$P_e(u, E^*, t, t_d)$	event probability of u encountering any $e \in E^*$ from t to t_d

DP users to upload data: 1) $\text{Greedy}_{\text{dp}}$ (cold-start) and 2) $\text{ExpectationBased}_{\text{dp}}$ (prediction-based).

1) $Greedy_{dp}$: $\text{Greedy}_{\text{dp}}$ follows the intuitions below to upload data.

a) If a DP user u_i encounters a server-related event, whether the event is making a call, encountering a Bluetooth gateway, or connecting to WiFi, u_i will upload data, because all these events cost less energy than creating a new 3G connection for data uploading.

b) If a DP user u_i encounters a user-related event, u_i will not offload data.

We use $E_{\text{dp\_like}}$ to represent the events described above which make the decision making true

$$E_{\text{dp\_like}} = \{e | e \in E_s, W_s(u, r, e) < W_{3G}(u, r)\}$$

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

2) $ExpectationBased_{dp}$: The intuition behind $\text{ExpectationBased}_{\text{dp}}$ is to compare the expected energy consumptions (i.e., expEnergy) needed for different data offloading schemes corresponding to possible events predicted before the end of each data uploading cycle (e.g., uploading data to the server or keeping data under a server-related event) and select the one with least expected energy consumption. In this process, $\text{ExpectationBased}_{\text{dp}}$ leverages users' mobility and call prediction to predict future events.

$\text{ExpectationBased}_{\text{dp}}$ might discard the current energy-efficient event to wait for another more energy-efficient event later, while $\text{Greedy}_{\text{dp}}$ always triggers the first-coming energy-efficient event, specifically, as follows.

a) When encountering the server-related events that consume less energy than 3G, $\text{Greedy}_{\text{dp}}$ will always make u_i upload data, while $\text{ExpectationBased}_{\text{dp}}$ will sometimes make u_i keep data. For example, if u_i currently makes a 3G call, and u_i is predicted to have a very high probability of meeting a Bluetooth gateway soon, then keeping data till the next event of encountering the Bluetooth gateway could be a better strategy in terms of energy saving (because uploading data via Bluetooth consumes less energy than piggybacking data on a call).

b) When encountering the user-related events, $\text{Greedy}_{\text{dp}}$ will always make u_i keep data, while $\text{ExpectationBased}_{\text{dp}}$ provides the possibility of offloading data between encountered DP users u_i and u_j. For example, if u_i has a much lower probability to upload data via energy-efficient methods (i.e., encounter $e \in E_{\text{dp\_like}}$) in the future than u_j, then offloading data from u_i to u_j can hopefully reduce the users' total energy consumption, because u_j later could probably upload data via an energy-efficient method.

Fig. 3 illustrates the basic decision making process of $\text{ExpectationBased}_{\text{dp}}$, where expEnergy refers to the user's total energy consumption predicted from the current event time (i.e., when u encounters e) to the end of uploading cycle (t_d). For example, suppose t_d is 12:00 and current time is 8:25. If user Bob makes a phone call, then we calculate Bob's expEnergy from 8:25 to 12:00 under two distinct

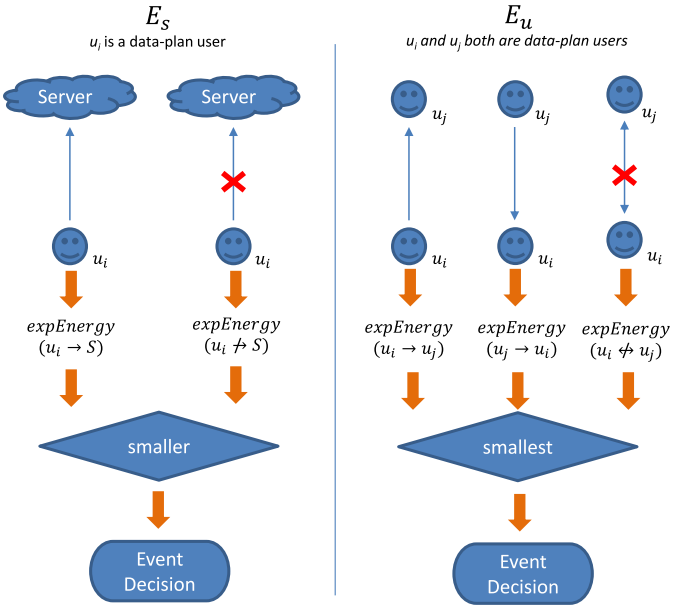


Fig. 3. Decision making process of $\text{ExpectationBased}_{\text{dp}}$.

conditions: uploading data over this call versus keeping data locally.

When encountering the server-related events, we calculate expEnergy for two possible conditions.

a) u_i uploads data to the server ($\text{expEnergy}_{u_i \to S}$).

b) u_i keeps data ($\text{expEnergy}_{u_i \not\to S}$).

And we select the scheme with smaller expEnergy.

When encountering the user-related events, if a DP user u_i meets a NDP user u_j, u_i will not offload data to u_j. (In fact, according to the NDP user data uploading schemes, the NDP user u_j will offload data to u_i if u_j has data.)

If a DP user u_i meets another DP user u_j, the decision making process is a little more complicated. We need to take both u_i and u_j's expected energy consumption into account. If we only consider one user's own energy consumption, u_i and u_j might both decide to send data to the counterpart, which leads to more energy consumption. We thus compute expEnergy for three possible conditions.

a) u_i offloads data to u_j ($\text{expEnergy}_{u_i \to u_j}$).

b) u_j offloads data to u_i ($\text{expEnergy}_{u_j \to u_i}$).

c) u_i and u_j both keep data ($\text{expEnergy}_{u_i \not\to u_j}$).

We choose the scheme with smallest expEnergy. To calculate the aforementioned expEnergy in Fig. 3, two basic components are involved.

The first component is the event energy consumption (defined in Section III, e.g., data transmission energy associated with 3G, WiFi, and Bluetooth). To estimate this part, we refer to existing work [13], [14]. Further details are shown in Section VI.

The second component is $\text{expEnergy}_{\text{keep}}(u, r, t, t_d)$, which represents a user u's expected energy consumption (from t to t_d) if u keeps the data of size r at time t. To compute $\text{expEnergy}_{\text{keep}}$, we first predict u's each event probability during $[t, t_d]$. Based on these event probabilities, we sum up all the event energy consumptions for offloading/uploading r. (Fig. 4 shows the detailed algorithm.) It is possible that

Problem:
Given a data-plan user u, local data (*size=r)* and time t, calculate the expected energy consumption during $[t, t_d]$ if u keeps data r at t.

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

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

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

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

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

Fig. 4. Calculation for $\text{expEnergy}_{\text{keep}}$.

$\text{expEnergy}_{\text{keep}}(u, 0, t, t_d) > 0$ (even though u holds no data at t), because u may receive data from other users during $[t, t_d]$ (see step 3 in Fig. 4).

With these two components, we can calculate all the aforementioned expEnergy in Fig. 3. The detailed formulas are given as follows.

1) *For $e \in E_s$:*
   a) *When u_i uploads data to the server ($expEnergy_{u_i\to S}$):* $\text{expEnergy}_{u_i\to S}$ includes two parts: i) the energy to upload data and ii) the expected energy consumption after this uploading (so u_i keeps no local data)

$$\text{expEnergy}_{u_i\to S} = W_s(u_i, r_i, e) + \text{expEnergy}_{\text{keep}}(u_i, 0, t, t_d).$$

   b) *When u_i keeps data ($expEnergy_{u_i\not\to S}$):* $\text{expEnergy}_{u_i\not\to S}$ includes only one part: the expected energy consumption when u_i still keeps data of size r_i

$$\text{expEnergy}_{u_i\not\to S} = \text{expEnergy}_{\text{keep}}(u_i, r_i, t, t_d).$$

2) *For $e \in E_u$*: For $e \in E_u$, we consider both u_i and u_j's energy consumptions together.
   a) *When u_i offloads data to u_j ($\text{expEnergy}_{u_i\to u_j}$):* $\text{expEnergy}_{u_i\to u_j}$ includes four parts: 1) u_i's energy consumption for sending data; 2) u_j's energy consumption for receiving data; 3) u_i's expected energy consumption after sending data (so u_i keeps no local data); and 4) u_j's expected energy consumption after receiving data (so u_j keeps both u_i's and u_j's data)

$$\text{expEnergy}_{u_i\to u_j} = W_{u\_sd}(u_i, r_i, e) + W_{u\_rv}(u_j, r_i, e) + \text{expEnergy}_{\text{keep}}(u_i, 0, t, t_d) + \text{expEnergy}_{\text{keep}}(u_j, r_i + r_j, t, t_d).$$

   b) *When u_j offloads data to u_i ($\text{expEnergy}_{u_j\to u_i}$):* similar to $\text{expEneryg}_{u_i\to u_j}$, we only need to exchange i and j in the above formula.
   c) *When u_i and u_j keep data ($\text{expEnergy}_{u_i\not\to u_j}$):* $\text{expEnergy}_{u_i\not\to u_j}$ includes two parts: both u_i's and u_j's expected energy consumption when they keep their own data (r_i and r_j)

$$\text{expEnergy}_{u_i\not\to u_j} = \text{expEnergy}_{\text{keep}}(u_i, r_i, t, t_d) + \text{expEnergy}_{\text{keep}}\big(u_j, r_j, t, t_d\big).$$

C. Scheme Selection

When a new user first participates in the crowdsensing task, since there is no activity history about this user, the cold-start scheme $\{\text{SimpleGreedy}_{\text{ndp}}, \text{Greedy}_{\text{dp}}\}$ is recommended. After a period of time (T_{change}), when the new user accumulates certain activity logs, he can change to the prediction-based schemes $\{\text{AdvancedGreedy}_{\text{ndp}}, \text{ExpectationBased}_{\text{dp}}\}$ to get better performance.

Deciding T_{change} is an important issue. The optimal T_{change} may vary from one specific mobile crowdsensing task to another. In this paper, we do not discuss how to choose T_{change} but focus on designing and evaluating the overall framework.

D. Additional Features

1) Information Exchange Between Users: When two users meet, they need to exchange some information with each other for running effSense. For example, when two DP users meet, one needs to know the value of $\text{expEnergy}_{\text{keep}}$ of the counterpart (for $\text{ExpectationBased}_{\text{dp}}$). We propose to exchange information between encountered users as follows: encoding all the event probabilities in the device name according to a predefined protocol. Then, when a user meets another one, each user will know the event probabilities that can be used to calculate the other user's $\text{expEnergy}_{\text{keep}}$. One defect of this solution is that the user's device name needs to change after a time period because the event probabilities change over time. Fortunately, this is a simple operation without much energy consumption.

2) Flexibility to User-Type Interchange: An interesting and valuable feature of effSense is that one user can change his user type at any time, so that he can decide whether he cares more about energy consumption or data cost. We can still use the previously proposed solution: encoding the user type in the device name. Then, one user could easily know the other user's type when they meet each other.

3) Mechanisms for Exit-Users: A user might exit the task during a uploading cycle, which means that the un-uploaded data in this exit-user's phone might fail to be uploaded to the server. Note that as we have the assumption of "offload and

TABLE II
ENERGY CONSUMPTION ESTIMATION OF 3G/BLUETOOTH

	Small-size(<10KB)	*Big-size(xKB)*
3G	12 J	12+0.025x J
Bluetooth	1 J	1+0.003x J

TABLE III
ENERGY CONSUMPTION ESTIMATION FOR CRITICAL EVENTS

	Small-size(<10KB)	*Big-size(xKB)*
e_{3g_call}[2]	$12 \times (1 - 75\%) = 3$ J	3+0.006x J
e_{bt_device}	1 J	1+0.003x J
e_{bt_user}[3]	2 J	2+0.006x J

dismiss," the un-uploaded data might include multiple users' data. To minimize the data loss incurred by exit-users, we propose the following mechanisms.

a) *Forced-Uploading (Known-Exit):* If effSense knows when a user exits from a crowdsensing task (e.g., a user instructs to exit from the mobile task app menu), then the app can upload/offload the un-uploaded data immediately (e.g., a DP user would initialize a new 3G connection).
b) *Notify-Reuploading (Sudden-Exit):* A user might exit suddenly (e.g., due to operating system errors). In this case, the mobile task app fails to operate normally for forced uploading. To relieve this problem, at the end of each uploading cycle, the server can use a method like dial-to-deliver [34] to notify the users whose data has not been uploaded, to upload/offload their data again. Thus, if a user u's data was relayed to a sudden-exit user but u does not exit, u can still have a chance to upload his data again in the end.

VI. EVALUATION

In this section, we evaluate the effSense framework using two real-world crowdsensing datasets: 1) MIT reality mining [8] and 2) Nodobo [15]. While the MIT reality mining dataset recorded more mobile users' call and mobility traces, the Nodobo dataset reported a more up-to-date mobile users' traces with different patterns.

A. Experimental Setup for the MIT Dataset

For the MIT dataset, we choose seven weeks of sensing data (2004.10.4-2004.11.21) from 71 active users. The 30 users who consumed the largest volume of mobile data are considered as DP users, and the remaining 41 users are NDP users—as the MIT data campaign provided 30 users with data plans subsidy [9]. We used the first five weeks of user data to build the model, and evaluated the performance of effSense using the data of the last two weeks. The data uploading cycle was set to one day, i.e., each uploading cycle starts at 00:00 and ends at 24:00. Thus, 14 rounds of data uploading occurred during the last two weeks.

This experiment involved three types of critical events.

1) e_{3g_call}: Making a 3G voice call.
2) e_{bt_device}: Encountering a Bluetooth gateway. Two Bluetooth gateways are in the experiment: localhost.media.mit.edu and studies.media.mit.edu.
3) e_{bt_user}: Encountering another user via Bluetooth.

Based on the literature about the mobile phone energy consumption [13], [14], we estimate the energy consumption for transmitting data through 3G and Bluetooth (Table II). Table III shows the estimation results of energy consumption for each critical event type.

B. Bluetooth Encounters

In this section, we address two practical issues related to the Bluetooth encounters in our experiment.

1) Bluetooth Contact Duration: Sometimes, the Bluetooth encounter between two users is too short to transfer all data successfully. As the MIT reality mining campaign only activates Bluetooth scanning every 5 min [9], we cannot accurately know how long two users really meet when they are in contact via Bluetooth. In the evaluation, we eliminated the short Bluetooth encounters that do not have enough time to transfer data between two devices as follows: we only use the encounters that can be discovered in two continuous 5-min scannings. For example, if user u_i meets user u_j at 12:00 via Bluetooth, we will use this encounter in our evaluation only if u_i can still meet u_j at 12:05. Due to the dataset limitation, u_i and u_j might meet each other just at the two time points of 12:00 and 12:05, while keeping away from each other between 12:00 and 12:05. However, with this data preprocessing about the device encounters, we believe that most of the device encounters could allow successful data transfer between devices via Bluetooth.

2) Bluetooth Scanning Energy: Note that the energy consumption of Bluetooth scanning is not considered in the evaluation due to two reasons.

a) Intermittent Bluetooth scanning is required by some crowdsensing tasks such as MIT reality mining and SociableSense [4]; so the energy consumption of Bluetooth scanning is not caused by data uploading, but is required by the crowdsensing task.
b) Bluetooth 4.0 low energy (BLE) technology is adopted by more and more up-to-date smartphones (iPhone 5s, Nexus 5, etc.).

With BLE, the battery drain of Bluetooth scanning is dramatically decreased [35]. Due to the energy efficiency of BLE, a lot of novel real-time smartphone sensing applications are emerging recently, which require the smartphone users to turn on Bluetooth and do intermittent scanning all the time (e.g., fitness sensing with FitBit wristbands[4]). Thus, we believe that in the near future, more smartphone users would like to have Bluetooth (with BLE) always on to support such novel applications; again the energy consumption of Bluetooth scanning is not caused by the data uploading of effSense, but is required by these applications.

[2]3G data transmission during call saves ∼75% energy [10].

[3]The energy consumption of $e_{\text{bt\_user}}$ is twice of $e_{\text{bt\_device}}$ because of one user sending and one user receiving.

[4]https://www.fitbit.com/

TABLE IV
ENERGY CONSUMPTION FOR DIFFERENT PHONE USAGES

Action	*Energy*
Idle (1 minute)	0.9 J
3G Call (1 minute)	75.9 J
SMS (1 message)	3.5 J
Bluetooth Scanning (20 seconds)	4.5 J

C. Energy Calculation and Battery Constraints

If a user's phone battery has already reached a low energy level, the user is typically not willing to relay data for other users. Thus, we prevent a user from relaying data when the user's phone battery level is lower than a predefined limit, e.g., 50% battery level.

Due to the lack of explicit battery information in the MIT dataset, we simulate a user phone's battery level based on real-time phone usage records, including calls, messages, mobile data usage, etc. Our simulation makes the following basic assumptions.

1) Each user's phone battery is fully charged as 100% level at 00:00 A.M. every day and the phone would not be charged during the day.
2) We adopted the basic energy consumption for each phone usage type using exiting statistics of Nokia N95 [14] (see Table IV). According to the specification, N95's full battery is 950 mAh/3.7 v, which means the total battery energy is $950 \times 0.001 \times 3.7 \times 3600 = 12654$ J.

We set the battery level limit to 50%. This limit is purposely set high for two reasons.

1) The phone usage records are not a complete set. Many application usage logs, such as games, are not included. The actual energy consumption should be higher than our simulation setting.
2) We set a high battery limit to ensure that relaying others' data will not bring significant inconvenience to phone users' own experiences.

D. Prediction of Critical Events

To estimate the critical event occurring probability $P_e(u, E^*, t_1, t_2)$, we use a Poisson distribution model [11], [12]. The computation process contains the following steps (note that E^* might include different kinds of events, e.g., $E^* = \{e_{3g\_call}, e_{bt\_device}\}$ represents the energy-efficient events for DP users).

Step 1: Split one week into 24×7 nonoverlapping timeslots—each timeslot lasts for 1 h; then, map t_1, t_2 to the corresponding timeslots in a week, i.e., ts_1 and ts_2, respectively.

Step 2: Assume the event $e_i \in E^*$ follows a Poisson process, then the probability of event e_i happening k times for user u during $[ts_1, ts_2]$ is

$$p(e_i, u, k, ts_1, ts_2) = \mu^k_{e_i,u,ts_1,ts_2} \cdot \exp\left(-\mu_{e_i,u,ts_1,ts_2}\right)/k!$$

where μ_{e_i,u,ts_1,ts_2} is estimated as the average number of the occurrences that user u encounters event e_i during $[ts_1, ts_2]$ from the history data.

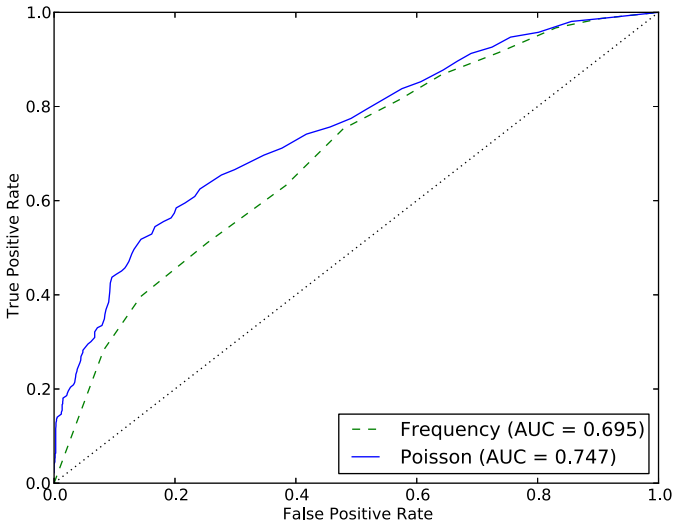


Fig. 5. ROC curve of call prediction (e_{3g_call}) on the MIT dataset.

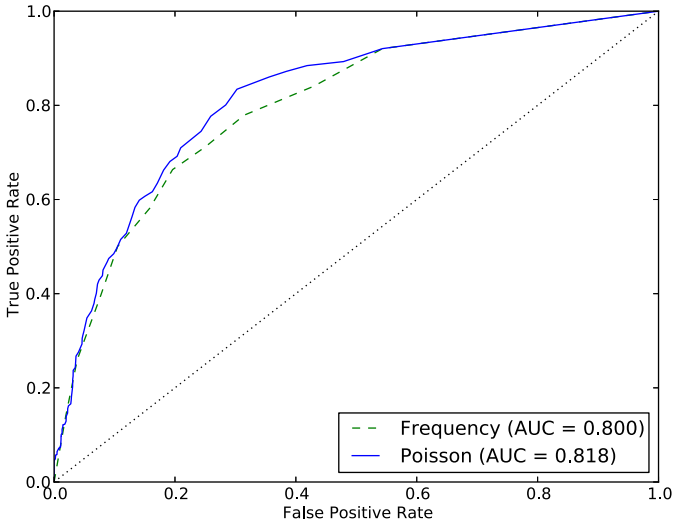


Fig. 6. ROC curve of Bluetooth encounter prediction (e_{bt_user}) on the MIT dataset.

Step 3: As $P_e(u, E^*, t_1, t_2)$ is the probability of at least one event $e_i \in E^*$ occurring at least once during $[t_1, t_2]$, we thus calculate it as follows:

$$P_e(u, E^*, t_1, t_2) = 1 - \prod_{e_i \in E^*} p(e_i, u, k = 0, ts_1, ts_2)$$
$$= 1 - \prod_{e_i \in E^*} \exp\left(-\mu_{e_i,u,ts_1,ts_2}\right).$$

To measure the performance of the Poisson method, we compare it with a simple method that directly counts the frequency of event occurrences, which is used in our previous work [36].

1) Frequency: Suppose that the history data includes m weeks, and there are n weeks u encounters any event $e_i \in E^*$ during $[ts_1, ts_2]$, then $P_e(u, E^*, t_1, t_2) = n/m$.

To compare the performance between the two methods, we draw receiver operating characteristics (ROC) curves [37] and calculate their area under the ROC curve (AUC) values.

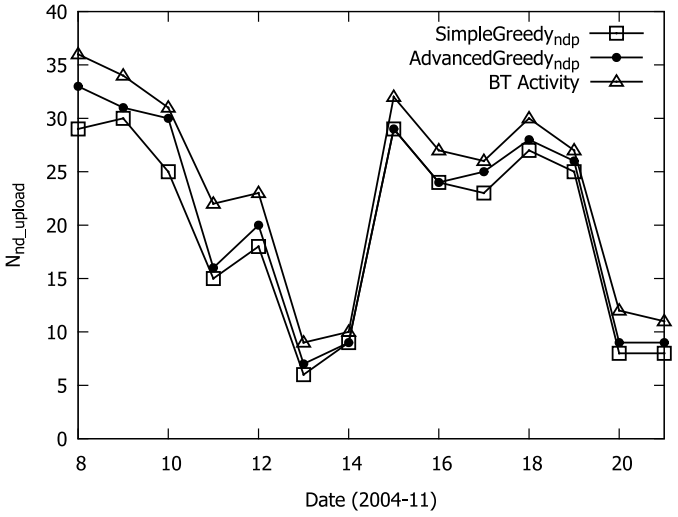


Fig. 7. N_{nd_upload} for two weeks on the MIT dataset.

Figs. 5 and 6 show the ROC curves associated to the prediction of 3G calls and Bluetooth encounters between users. We can see that the Poisson method outperforms the Frequency method in both cases. Thus, different from our previous work [36], we use the Poisson method in all the evaluations.

E. Experimental Results on the MIT Dataset

In order to evaluate the performance of proposed data uploading schemes in effSense, we design the experiments to address the following key questions.

1) *Data Cost:* How many NDP users can upload their data before the uploading cycle deadline without incurring data cost?
2) *Energy Consumption:* How much phone energy is consumed for DP users during the data uploading process?
3) *Event Triggering:* How many critical events have triggered data uploading/offloading during the data uploading process?

1) Data Cost Conservation: First, we investigate the performance of two effSense schemes for NDP users—the number of NDP users who upload data successfully in each data uploading cycle (noted as N_{nd_upload}). Fig. 7 plots the detailed N_{nd_upload} in two weeks for the two schemes, together with the upper-bound of N_{nd_upload}—the number of NDP users having Bluetooth activities (BT activity). This is because only Bluetooth activities can trigger successful data uploading for NDP users without data cost. In Fig. 7, effSense does not perform very well in weekends (11/13, 11/14, 11/20, 11/21) and Veterans day (11/11), as few users came to school so that the opportunities for Bluetooth relay dropped.

In Table V, we further list the statistics, observing that effSense helps on average 19.7 NDP users using SimpleGreedy$_{ndp}$, and 21.2 NDP users using AdvancedGreedy$_{ndp}$, corresponding to the success rate of 48% and 52% for 41 NDP users, respectively. AdvancedGreedy$_{ndp}$ outperforms SimpleGreedy$_{ndp}$ by 4%. This improvement is quite significant for AdvancedGreedy$_{ndp}$, as SimpleGreedy$_{ndp}$ has already achieved more than 85% of the upper bound BT activity on weekdays.

TABLE V
AVERAGE N_{nd_upload} IN ONE DATA UPLOADING CYCLE ON THE MIT DATASET (VALUES IN THE BRACKETS ARE THE PROPORTIONS TO THE VALUE OF BT ACTIVITY)

	Weekday	*Weekend*	*Overall*
BT Activity	28.8	10.5	23.6
SimpleGreedy	24.5 (85.7%)	7.8 (74.3%)	19.7 (83.5%)
AdvancedGreedy	26.2 (91.0%)	8.5 (80.6%)	21.2 (89.8%)

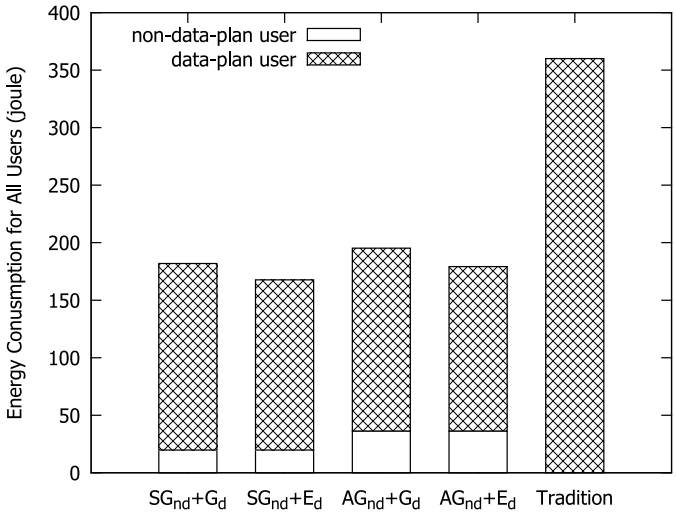


Fig. 9. totalEnergy$_{all}$ for small-size data on the MIT dataset.

2) Energy Conservation: Fig. 8 shows the energy consumption for DP users in one data uploading cycle (noted as totalEnergy$_{dp}$) when three different types of sensed data sizes are considered—small (1 kB), 100 kB, and 200 kB. effSense reduces 55%–65% of the energy consumption for DP users compared with the traditional method.[6] When using the same NDP scheme (either SimpleGreedy$_{ndp}$ or AdvancedGreedy$_{ndp}$), ExpectationBased$_{dp}$ can save up to about 13% extra energy for DP users compared with Greedy$_{dp}$ on weekdays. On weekends, the energy consumption difference between ExpectationBased$_{dp}$ and Greedy$_{dp}$ is not significant, because few students go to school on weekends and thus data relays between DP users will rarely happen. 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.

Fig. 9 shows the total energy consumption for all DP and NDP users (totalEnergy$_{all}$). Though effSense causes NDP users to consume energy, which does not exist in the traditional method, totalEnergy$_{all}$ is still reduced by 46%–54%.

3) Event Triggering: Fig. 10 shows the total number of events that trigger data uploading/offloading for each event type, and Fig. 11 shows the corresponding triggering probability. Here, the events are 3G-call, BT-device, and BT-user.

[6]The traditional method means that DP users upload their data via a new 3G connection every day, while NDP users store their data in the SD-cards instead of uploading to the server. So, NDP users will not consume energy in data uploading.

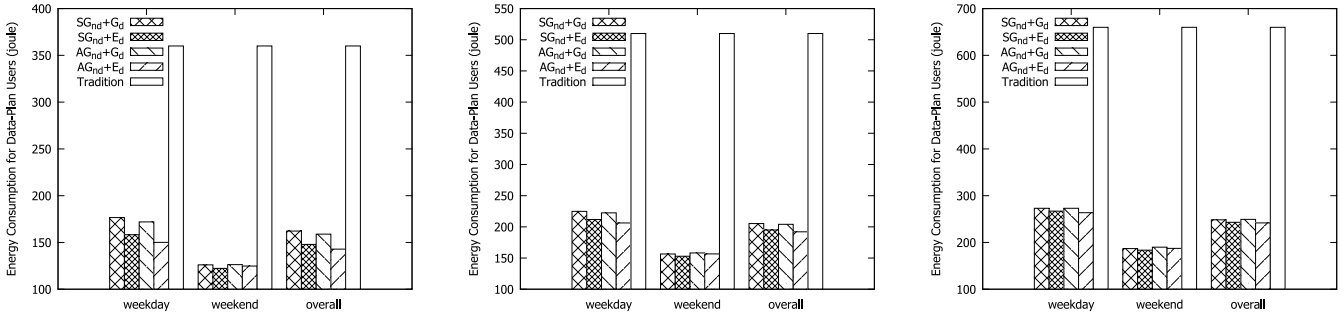


Fig. 8. Energy consumption of DP users, totalEnergy$_{dp}$, for uploading (a)–(c) small(1 kB)/100 kB/200 kB data per cycle on the MIT dataset (SG_{nd}: SimpleGreedy$_{ndp}$; AG_{nd}: AdvancedGreedy$_{ndp}$; G_d: Greedy$_{dp}$; E_d: ExpectationBased$_{dp}$).

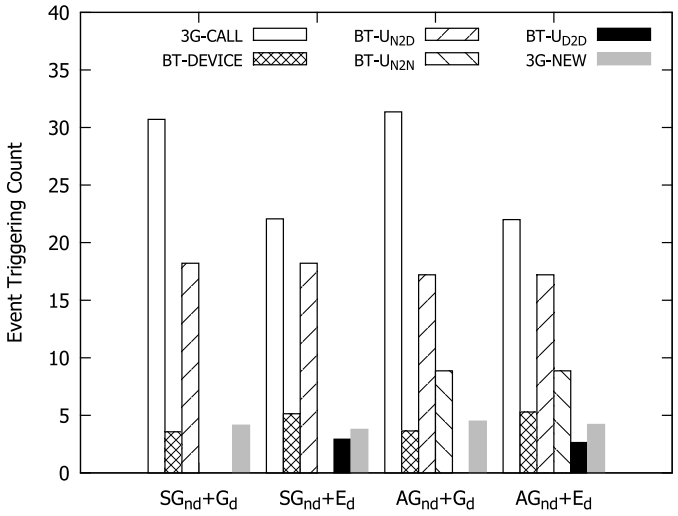


Fig. 10. Event triggering per uploading cycle on the MIT dataset (small data).

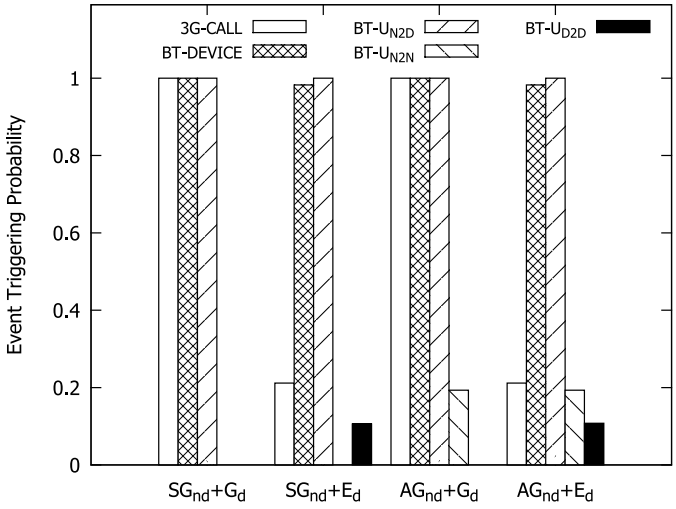


Fig. 11. Event triggering probability per uploading cycle on the MIT dataset (small data).

In particular, BT-user can be further divided into three categories according to user type: 1) NDP → DP (BT_U_{N2D}); 2) NDP → NDP (BT_U_{N2N}); and 3) DP → DP (BT_U_{D2D}).[7]

First, the frequency of requiring a new 3G connection decreases significantly in effSense compared with the traditional method that would incur 30 new 3G connections in each data uploading cycle. As establishing a new 3G connection is energy-demanding, this is the primary reason why effSense can reduce energy consumption significantly. In addition, for the cold-start scheme pair {SimpleGreedy$_{ndp}$, Greedy$_{dp}$}, the data exchanges between users are only carried out when NDP → DP. However, AdvancedGreedy$_{ndp}$ and ExpectationBased$_{dp}$ introduce the data relays at NDP → NDP and DP → 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 further conserve energy compared with Greedy$_{dp}$, as the energy consumption of e_{bt_device} is less than e_{3g_call}.

4) Impact of Exit-Users: Exit-users can incur data loss and thus effSense needs some mechanisms to deal with exit-users (Section V-D3). Here we evaluate how many users' sensed data would fail to be uploaded (N_{fail}) due to exit-users in one uploading cycle by simulations.

[7]DP → NDP does not exist in effSense because this direction conflicts with the goal of saving data cost for NDP users.

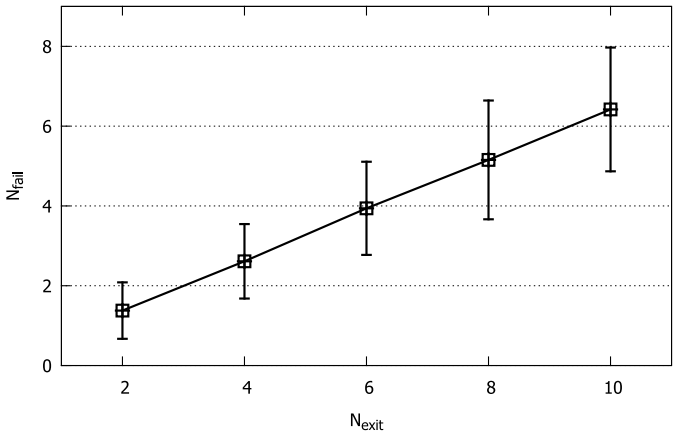


Fig. 12. N_{fail} for different N_{exit} on the MIT dataset.

a) Exit-model: We randomly choose N_{exit} users to be exit-users (either known-exit or sudden-exit). For each selected user, we randomly assign a time line within the delayed uploading cycle as his exit timing, via uniform distribution.

TABLE VI
RESULTS OF DIFFERENT DP/NDP RATIOS (24-HOUR DELAY)

$\|U_{dp}\|$	$\|U_{ndp}\|$	$N_{nd\_upload}/\|U_{ndp}\|$	$totalEnergy_{dp}/\|U_{dp}\|$
10	61	38%	6.06 J
20	51	47%	5.23 J
30	41	51%	4.76 J
40	31	53%	4.40 J
50	21	57%	4.26 J
60	11	68%	4.05 J

TABLE VII
RESULTS FOR DIFFERENT MAX DELAYS $d_{\max}$ (30 DP USERS)

d_{max}	N_{nd_upload}	$totalEnergy_{dp}$
3-hour	13.3	206.1 J
6-hour	17.0	184.4 J
12-hour	19.0	152.5 J
24-hour	21.2	142.9 J

TABLE VIII
RESULTS FOR DIFFERENT START TIME (30 DP USERS, 3-HOUR DELAY)

Start	N_{nd_upload}	$totalEnergy_{dp}$
8:00	7.2	189.4 J
11:00	14.3	209.1 J
14:00	13.2	195.4 J
17:00	10.8	181.4 J

Fig. 12 shows N_{fail} and its standard deviation, varying with N_{exit} (2–10) based on 100 simulations. Generally, N_{fail} is less than N_{exit}, which means that in most simulations, some exit-users have already uploaded/offloaded their sensed data before they exit. The ratio $N_{\text{fail}}/N_{\text{exit}}$ is around 65%, no distinction between different N_{exit}. Therefore, if N_{exit} users exit in one cycle, the expected N_{fail} is about $0.65N_{\text{exit}}$, less than N_{exit}. We also run the simulations when effSense does not deploy the exit-user mechanisms described in Section V-D3, and the ratio $N_{\text{fail}}/N_{\text{exit}}$ is larger (75%–80%). This verifies the effectiveness of our proposed exit-user mechanisms.

F. Parametric Analysis on the MIT Dataset

In this section, we analyze the two key parameters in effSense and study how they affect the performance of the framework. The parameters are DP/NDP user ratio and max tolerable delay ($d_{\max}$). Our experiments here focus on the scheme pair {AdvancedGreedy$_{\text{ndp}}$, ExpectationBased$_{\text{dp}}$} and small sensed data size.

1) DP/NDP User Ratio: In the previous experiments, we selected 30 out of 71 participants as DP users, as this is the actual setting in the MIT reality mining project [9]; thus, the DP/NDP ratio is 30/41. Real-life crowdsensing tasks face various DP/NDP ratios. Furthermore, in a long period of crowdsensing, existing participants could leave and new participants could join. The objective of evaluations here is to verify effSense's performance with different DP/NDP ratios. Instead of investigating the absolute N_{nd_upload} and totalEnergy$_{\text{dp}}$ metrics, we investigate their relative values to better present effSense's robustness. The two relative metrics are: $N_{nd\_\text{upload}}/|U_{\text{ndp}}|$ (the percentage of NDP users who upload data successfully) and totalEnergy$_{\text{dp}}/|U_{\text{dp}}|$ (the average energy consumption of each DP user).

Table VI shows the evaluation results under six different DP/NDP user ratio settings, and from the table we have the following observations.

a) With more DP users, greater percentage of NDP users can upload data successfully without incurring data cost (i.e., $N_{nd\_\text{upload}}/|U_{\text{ndp}}|$ increases), because NDP users could have more chances to encounter DP users and relay data to them.
b) With more DP users, average energy consumption for each DP user decreases (i.e., totalEnergy$_{\text{dp}}/|U_{\text{dp}}|$ decreases), because each DP user needs to help fewer NDP users to relay data.

Though effSense performs better with more DP users, it also works well when only few DP users exist. As shown in Table VI, even under low $|U_{\text{dp}}|/|U_{\text{ndp}}|$ such as 10/61, nearly 40% of the NDP users could upload data successfully without data cost, and a DP user usually consumes less than 50% energy on average, compared with uploading data by creating a new 3G connection which consumes 12 J energy.

2) Max Tolerable Delay: All the evaluation results reported so far were carried out assuming the max tolerable delay ($d_{\max}$) for data uploading is 24 h, as the MIT reality mining project asked DP users to upload data once a day [9]. Nowadays, 24 h is a long uploading delay as users most likely be able to upload data using free WiFi at home. Thus, we conduct extra experiments to test the performance at smaller $d_{\max}$ (see Table VII). As $d_{\max}$ decreases, NDP users have less chance to upload data without incurring data cost (i.e., N_{nd_upload} decreases) and DP users consume more energy (i.e., totalEnergy$_{\text{dp}}$ increases) in each data uploading cycle. This is because with a shorter delay, there are fewer critical events occurring for effSense to trigger data offloading in order to reduce data cost and/or energy consumption.

In addition, we observe that the start time of data uploading cycle affects N_{nd_upload} and totalEnergy$_{\text{dp}}$ when applying smaller $d_{\max}$, as the number of each type of critical events differ greatly between different time periods. Table VIII shows the evaluation results with different data uploading cycle start time and a max tolerable delay of 3 h. N_{nd_upload} is higher in the afternoon than in the morning/evening, as NDP users are more likely to encounter DP users to relay in the afternoon; totalEnergy$_{\text{dp}}$ is lower in the evening than in the morning/afternoon, as users are likely to make calls in the evening.

G. Evaluation on the Nodobo Dataset

We also evaluated effSense on the Nodobo dataset with users' WiFi traces. It contains a type of critical event, e_{wifi}, different from the MIT dataset, when a user connects to a WiFi AP. Free WiFi APs can be used for NDP users to upload data without incurring data cost. To simplify the experiment, we assume all WiFi APs in the Nodobo project are free. In the Nodobo project, there is no Bluetooth gateway, so we excluded $e_{\text{bt\_device}}$ events, and focus on the e_{wifi} events.

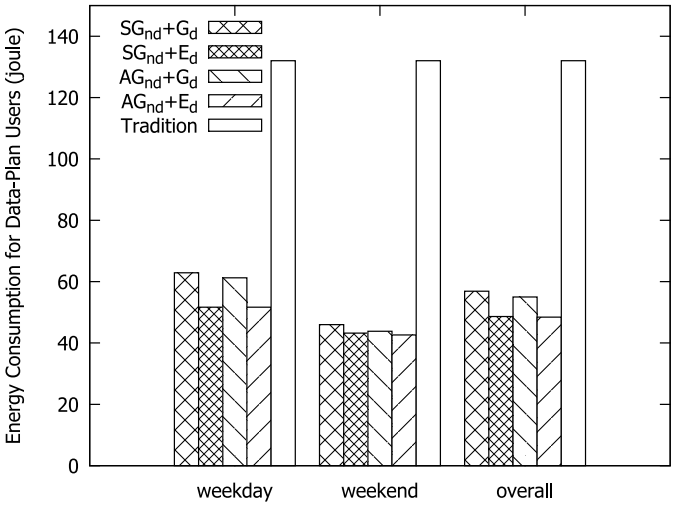


Fig. 13. $\text{totalEnergy}_{\text{dp}}$ for small-size data on the Nodobo dataset.

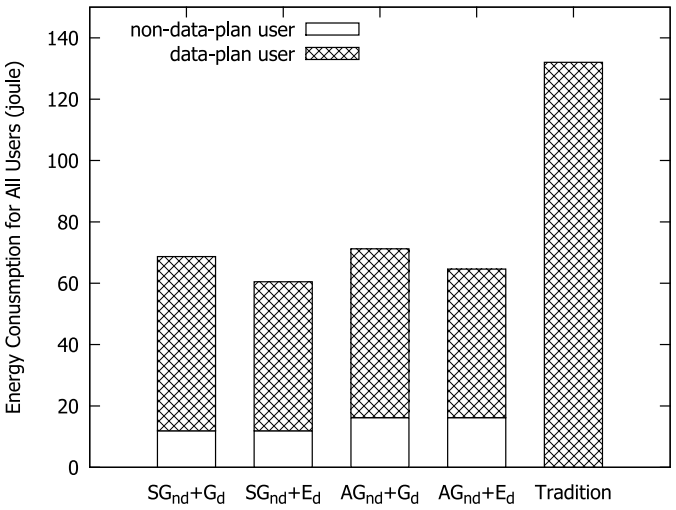


Fig. 14. $\text{totalEnergy}_{\text{all}}$ for small-size data on the Nodobo dataset.

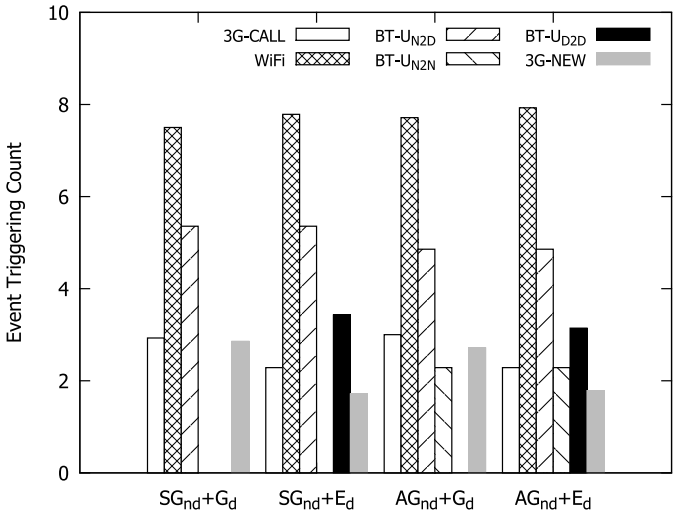


Fig. 15. Event triggering per uploading cycle on the Nodobo dataset.

TABLE IX
N_{nd_upload} ON THE NODOBO DATASET (VALUES IN THE BRACKETS ARE THE PROPORTIONS TO THE CORRESPONDING VALUE OF BT/WIFI ACTIVITY)

	Weekday	*Weekend*	*Overall*
BT/WiFi Activity	10.3	7.2	9.2
SimpleGreedy	10.0 (96.8%)	6.2 (86.1%)	8.6 (93.8%)
AdvancedGreedy	10.3 (100%)	6.4 (88.9%)	8.9 (96.9%)

According to [13], uploading small-size data via WiFi consumes 2 J energy. In the experiment of the Nodobo dataset, we set the same $d_{\max}$ (i.e., 24 h), and DP/NDP ratio as 11/16 (similar to 30/41 on the MIT dataset).

The results are shown in Figs. 13–15, and Table IX (we set the upper bound of N_{nd_upload} as the number of NDP users who have either Bluetooth or WiFi activity, called BT/WiFi activity). Like the previous experiments on the MIT dataset, these results show similar performance in conserving both data cost and energy consumption.

Fig. 15 shows that e_{wifi} plays an active role in improving effSense's performance as it frequently triggers data uploading to conserve data cost or energy consumption. Note that N_{nd_upload} can reach the upper bound when using $\text{AdvancedGreedy}_{\text{ndp}}$ scheme for NDP users in weekdays (see Table IX). These results show the flexibility and extensibility of the effSense framework, and reveal the potential that effSense can become more powerful when introducing more critical events.

VII. DISCUSSION

A. Data Size

In the experiment, we set three data sizes—small (1 kB), 100 kB, and 200 kB, which are not large, and thus our assumption of "offload all data" can probably be satisfied. However, some mobile crowdsensing tasks can generate sensed data in a large volume up to several MB (e.g., taking photos). When the data size is large, the assumption of offload all data will become a bit unrealistic, especially for the events such as Bluetooth encounters (often temporary and unstable). In the future, effSense can be improved to handle sensing a large volume of data.

B. User Incentives for Data Relay

Data relay between users is important in effSense to help reduce data cost and energy consumption. In the experiment, we assumed that users are always willing to help others relay data when their phone battery is not low (>50%). However, this is not always true in real life; effSense in the future could additionally design incentive mechanisms for encouraging users with the sufficient battery life to actively relay data for others.

C. Diversity of Critical Events

Although we identified several useful critical events in this paper, there are still other critical events that we could consider in our future work. One example is the event of charging the mobile phone battery. When a user's phone is charging, energy consumption of data uploading could be

assumed to be zero. Because of the dataset limitation, we cannot include these critical events in our current evaluation. However, effSense is flexible to include new critical events in the framework. Like these existing critical events, adding a new event in effSense should address two key practical issues, i.e., estimating energy consumption of uploading/offloading data under such an event, and providing the model for event prediction.

D. Delay-Tolerant and Real-Time Tasks

The main purpose of effSense is to support delay-tolerant data uploading, but it can be integrated with real-time data uploading to better serve application requirements. For example, users use effSense to deal with delay-tolerant crowdsensing tasks, and initialize new 3G connections (or WiFi if possible) to upload data for real-time crowdsensing tasks. Although it seems like a simple integration, this solution can further improve the performance of effSense. For example, if a DP user participates in both delay-tolerant and real-time crowdsensing tasks, then he regularly uploads real-time sensed data at certain time every day. effSense can leverage this routine uploading event to reduce his energy consumption by uploading the delay-tolerant data simultaneously, because uploading multiple data together to the server can reduce the transmission energy overhead [27].

E. Prediction of Event Probability

We use a Poisson distribution model to predict the critical event probability. As this paper does not focus on the mobility/activity prediction, we apply this state-of-the-art method [11], [12] and focus on the data uploading schemes to show the feasibility and effectiveness of effSense. A more advanced prediction method may further enhance the framework. However, when trying a complex prediction method, it is inevitable to consider that the prediction itself could consume a certain amount of energy.

F. User Data Privacy

User data privacy is an important concern during the process of data relay. Some approaches [38] have been proposed in the delay-tolerant network to ensure anonymity and security. In the future, we will study the possibility to incorporate such security mechanisms into effSense to protect user privacy.

VIII. Conclusion

In this paper, we investigate the problem of how to minimize both energy consumption and data cost caused by data uploading in mobile crowdsensing. We address the key concerns for both DP and NDP users (energy consumption versus data cost) simultaneously, and design a data uploading framework called effSense to improve both types of users' experience in mobile crowdsensing tasks.

For NDP users, we propose to reduce or eliminate data cost in data uploading by using the least expensive network (e.g., Bluetooth) other than 3G. For DP users, we propose to choose the appropriate critical events (e.g., making a voice call) to trigger data uploading that requires less energy rather than create a new 3G connection. We design two dedicated schemes for both user types to help users make proper decisions about whether to offload or keep data when encountering a critical event. Our effSense framework was evaluated with the MIT and Nodobo datasets. The experiment results verified the efficiency and effectiveness of effSense for data uploading in mobile crowdsensing tasks.

Acknowledgment

The authors would like to thank the reviewers and the editors for their helpful comments and suggestions. The authors would also like to thank C. Chen, D. Yang, X. Han, and B. Y. Lim for insightful discussions and careful proofreading.

References

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

[2] D. Zhang, L. Wang, H. Xiong, and B. Guo, "4w1h in mobile crowd sensing," *IEEE Commun. Mag.*, vol. 52, no. 8, pp. 42–48, Aug. 2014.

[3] R. K. Rana, C. T. Chou, S. S. Kanhere, N. Bulusu, and W. Hu, "Ear-phone: An end-to-end participatory urban noise mapping system," in *Proc. 9th ACM/IEEE Int. Conf. Inf. Process. Sensor Netw. (IPSN)*, Stockholm, Sweden, 2010, pp. 105–116.

[4] K. K. Rachuri, C. Mascolo, M. Musolesi, and P. J. Rentfrow, "SociableSense: Exploring the trade-offs of adaptive sampling and computation offloading for social sensing," in *Proc. 17th Annu. Int. Conf. Mobile Comput. Netw. (MobiCom)*, Las Vegas, NV, USA, 2011, pp. 73–84.

[5] W. Sherchan *et al.*, "Using on-the-move mining for mobile crowdsensing," in *Proc. IEEE 13th Int. Conf. Mobile Data Manage. (MDM)*, Bangalore, India, 2012, pp. 115–124.

[6] M. Musolesi, M. Piraccini, K. Fodor, A. Corradi, and A. T. Campbell, "Supporting energy-efficient uploading strategies for continuous sensing applications on mobile phones," in *Proc. 8th Int. Conf. Pervas. Comput.*, Helsinki, Finland, 2010, pp. 355–372.

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

[8] N. Eagle and A. (Sandy) Pentland, "Reality mining: Sensing complex social systems," *Pers. Ubiquit. Comput.*, vol. 10, no. 4, pp. 255–268, 2006.

[9] N. Eagle. (Sep. 10, 2014). *The Reality Mining Data Readme*. [Online]. Available: http://www.media.mit.edu/ventures/EPROM/data/RealityMining_ReadMe.pdf

[10] J. Nurminen, "Parallel connections and their effect on the battery consumption of a mobile phone," in *Proc. 7th IEEE Consum. Commun. Netw. Conf. (CCNC)*, Las Vegas, NV, USA, 2010, pp. 1–5.

[11] R. I. Ciobanu and C. Dobre, "Predicting encounters in opportunistic networks," in *Proc. 1st ACM Workshop High Perform. Mobile Opportunistic Syst. (HP-MOSys)*, Paphos, Cyprus, 2012, pp. 9–14.

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

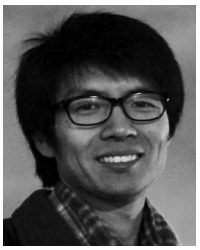
[13] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "Energy consumption in mobile phones: A measurement study and implications for network applications," in *Proc. 9th ACM SIGCOMM Conf. Internet Meas. Conf. (IMC)*, Chicago, IL, USA, 2009, pp. 280–293.

[14] G. Perrucci, F. Fitzek, and J. Widmer, "Survey on energy consumption entities on the smartphone platform," in *Proc. IEEE 73rd Veh. Technol. Conf. (VTC)*, Yokohama, Japan, 2011, pp. 1–6.

[15] S. Bell, A. McDiarmid, and J. Irvine, "Nodobo: Mobile phone as a software sensor for social network research," in *Proc. IEEE 73rd Veh. Technol. Conf. (VTC)*, Yokohama, Japan, 2011, pp. 1–5.

[16] S. Reddy, D. Estrin, and M. Srivastava, "Recruitment framework for participatory sensing data collections," in *Proc. 8th Int. Conf. Pervas. Comput.*, Helsinki, Finland, 2010, pp. 138–155.

[17] G. Cardone *et al.*, "Fostering participaction in smart cities: A geo-social crowdsensing platform," *IEEE Commun. Mag.*, vol. 51, no. 6, pp. 112–119, Jun. 2013.
[18] M.-R. Ra, B. Liu, T. F. La Porta, and R. Govindan, "Medusa: A programming framework for crowd-sensing applications," in *Proc. 10th Int. Conf. Mobile Syst. Appl. Serv. (MobiSys)*, Low Wood Bay, U.K., 2012, pp. 337–350.
[19] G. Cardone, A. Cirri, A. Corradi, and L. Foschini, "The participact mobile crowd sensing living lab: The testbed for smart cities," *IEEE Commun. Mag.*, vol. 52, no. 10, pp. 78–85, Oct. 2014.
[20] D. Zhang, H. Xiong, L. Wang, and G. Chen, "CrowdRecruiter: Selecting participants for piggyback crowdsensing under probabilistic coverage constraint," in *Proc. ACM Int. Joint Conf. Pervas. Ubiquit. Comput. (UbiComp)*, Seattle, WA, USA, 2014, pp. 703–714.
[21] H. Xiong, D. Zhang, L. Wang, and H. Chaouchi, "EMC3: Energy-efficient data transfer in mobile crowdsensing under full coverage constraint," *IEEE Trans. Mobile Comput.*, vol. 14, no. 7, pp. 1355–1368, Jul. 2015.
[22] G. Cohn *et al.*, "An ultra-low-power human body motion sensor using static electric field sensing," in *Proc. ACM Conf. Ubiquit. Comput. (UbiComp)*, Pittsburgh, PA, USA, 2012, pp. 99–102.
[23] M. B. Kjrgaard, S. Bhattacharya, H. Blunck, and P. Nurmi, "Energy-efficient trajectory tracking for mobile devices," in *Proc. 9th Int. Conf. Mobile Syst. Appl. Serv. (MobiSys)*, Washington, DC, USA, 2011, pp. 307–320.
[24] Y. Chon, E. Talipov, and H. Cha, "Autonomous management of everyday places for a personalized location provider," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 42, no. 4, pp. 518–531, Jul. 2012.
[25] M.-R. Ra, B. Priyantha, A. Kansal, and J. Liu, "Improving energy efficiency of personal sensing applications with heterogeneous multi-processors," in *Proc. ACM Conf. Ubiquit. Comput. (UbiComp)*, Pittsburgh, PA, USA, 2012, pp. 1–10.
[26] E. Cuervo *et al.*, "MAUI: Making smartphones last longer with code offload," in *Proc. 8th Int. Conf. Mobile Syst. Appl. Serv. (MobiSys)*, San Francisco, CA, USA, 2010, pp. 49–62.
[27] N. D. Lane *et al.*, "Piggyback crowdsensing (PCS): Energy efficient crowdsourcing of mobile sensor data by exploiting smartphone app opportunities," in *Proc. 11th ACM Conf. Embedded Netw. Sensor Syst. (SenSys)*, Rome, Italy, 2013, pp. 7:1–7:14.
[28] M.-R. Ra *et al.*, "Energy-delay tradeoffs in smartphone applications," in *Proc. 8th Int. Conf. Mobile Syst. Appl. Serv. (MobiSys)*, San Francisco, CA, USA, 2010, pp. 255–270.
[29] P. Bellavista, A. Corradi, and C. Giannelli, "The real ad-hoc multi-hop peer-to-peer (RAMP) middleware: An easy-to-use support for spontaneous networking," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Riccione, Italy, 2010, pp. 463–470.
[30] B. Hull *et al.*, "CarTel: A distributed mobile sensor computing system," in *Proc. 4th Int. Conf. Embedded Netw. Sensor Syst. (SenSys)*, Boulder, CO, USA, 2006, pp. 125–138.
[31] E. Soroush, K. Wu, and J. Pei, "Fast and quality-guaranteed data streaming in resource-constrained sensor networks," in *Proc. ACM Symp. Mobile Ad Hoc Netw. Comput. (MobiHoc)*, Hong Kong, 2008, pp. 391–400.
[32] L. Vu, Q. Do, and K. Nahrstedt, "Jyotish: A novel framework for constructing predictive model of people movement from joint WiFi/Bluetooth trace," in *Proc. IEEE Int. Conf. Pervas. Comput. Commun. (PerCom)*, Seattle, WA, USA, 2011, pp. 54–62.
[33] S. Phithakkitnukoon, R. Dantu, R. Claxton, and N. Eagle, "Behavior-based adaptive call predictor," *ACM Trans. Auton. Adapt. Syst.*, vol. 6, no. 3, pp. 21:1–21:28, 2011.
[34] L. Ravindranath, A. Thiagarajan, H. Balakrishnan, and S. Madden, "Code in the air: Simplifying sensing and coordination tasks on smartphones," in *Proc. 12th Workshop Mobile Comput. Syst. Appl. (HotMobile)*, San Diego, CA, USA, 2012, pp. 4:1–4:6.
[35] C. Gomez, J. Oller, and J. Paradells, "Overview and evaluation of Bluetooth low energy: An emerging low-power wireless technology," *Sensors*, vol. 12, no. 9, pp. 11734–11753, 2012.
[36] L. Wang, D. Zhang, and H. Xiong, "effSense: Energy-efficient and cost-effective data uploading in mobile crowdsensing," in *Proc. ACM Conf. Pervas. Ubiquit. Comput. Adjunct Publ. (UbiComp Adjunct)*, Zurich, Switzerland, 2013, pp. 1075–1086.
[37] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognit. Lett.*, vol. 27, no. 8, pp. 861–874, 2006.
[38] A. Kate, G. M. Zaverucha, and U. Hengartner, "Anonymity and security in delay tolerant networks," in *Proc. 3rd Int. Conf. Secur. Privacy Commun. Netw. Workshops (SecureComm)*, Nice, France, 2007, pp. 504–513.



Leye Wang received the B.Sc. and M.Sc. degrees in computer science from Peking University, Beijing, China, in 2009 and 2012, respectively. He is currently pursuing the Ph.D. degree with Institut Mines-TELECOM/TELECOM SudParis, Évry, France, and Université Pierre-et-Marie-Curie, Paris, France.

His current research interests include mobile crowd sensing and ubiquitous computing.

Daqing Zhang (M'11) received the Ph.D. degree from the University of Rome "La Sapienza," Rome, Italy, and the University of L'Aquila, L'Aquila, Italy, in 1996.

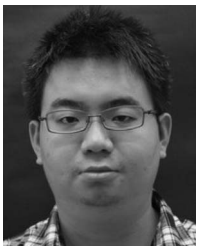
He is a Professor with the Institut Mines-TELECOM/TELECOM SudParis, Évry, France. His current research interests include large-scale data mining, urban computing, context-aware computing, and ambient assistive living. He has published over 180 referred journal and conference papers. All his research has been motivated by practical applications in digital cities, mobile social networks, and elderly care.

Dr. Zhang was a recipient of the Ten Years CoMoRea Impact Paper Award at the IEEE International Conference on Pervasive Computing and Communications 2013, the Best Paper Award at the IEEE International Conference on Ubiquitous Intelligence and Computing 2012, and the Best Paper Runner Up Award at Mobiquitous 2011. He is an Associate Editor of four journals, including *ACM Transactions on Intelligent Systems and Technology*. He has been a frequent Invited Speaker in various international events on ubiquitous computing.

Zhixian Yan (M'12) received the Ph.D. degree in computer sciences from the Swiss Federal Institute of Technology, Lausanne, Switzerland, in 2011.

He was a Research Staff with Samsung Research America in Silicon Valley, San Jose, CA, USA. His current research interests include mobile sensing, wearable computing, data management, machine learning, data mining, data science, and real-world big-data applications. He has published over 30 peer-reviewed papers.

Dr. Yan has received several best paper awards and nominations, including MobiDE 2012, ISWC 2012, and MDM 2012.



Haoyi Xiong (M'15) received the B.Eng. degree in electrical engineering from the Huazhong University of Science and Technology, Wuhan, China, the M.Sc. degree in information technology from the Hong Kong University of Science and Technology, Hong Kong, and the Ph.D. degree (Highest Hons.) in computer science from Université Pierre et Marie Curie, Paris, France, and TELECOM SudParis, Évry, France, in 2009, 2010, and 2015, respectively.

He is currently a Postdoctoral Research Fellow with the Institut Mines-TELECOM, Évry, France. His current research interests include mobile crowd-sensing and human mobility modeling and analysis.

Dr. Xiong was a recipient of the best paper award from the IEEE International Conference on Ubiquitous Intelligence and Computing in 2012.

Bing Xie was born in Hunan, China, in 1970. He received the Ph.D. degree in computer science from the Changsha Institute of Technology, Hunan, in 1998.

He held a postdoctoral position with Peking University, Beijing, China, in 1998, where he was an Associate Professor until 2006. He is currently a Professor with the Department of Computer Science, Peking University. His current research interests include software reuse, ubiquitous computing, and its industrial application. He has authored or co-authored over 60 international publications. He has hosted nine research projects (as Principal Investigator) supported by the Chinese Government concerning software engineering and formal methods.