Let Opportunistic Crowdsensors Work Together for Resource-efficient, Quality-aware Observations

Yifan Du
Inria Paris, France
yifan.du@inria.fr

Françoise Sailhan
CNAM Paris, France
francoise.sailhan@cnam.fr

Valérie Issarny
Inria Paris, France
valerie.issarny@inria.fr

***Abstract*—Opportunistic crowdsensing empowers citizens carrying hand-held devices to sense physical phenomena of common interest at a large and fine-grained scale without requiring the citizens' active involvement. However, the resulting uncontrolled collection and upload of the massive amount of contributed raw data incur significant resource consumption, from the end device to the server, as well as challenge the quality of the collected observations. This paper tackles both challenges raised by opportunistic crowdsensing, that is, enabling the resource-efficient gathering of relevant observations. To achieve so, we introduce the *BeTogether* middleware fostering context-aware, collaborative crowdsensing at the edge so that co-located crowdsensors operating in the same context, group together to share the workload in a cost- and quality-effective way. We evaluate the proposed solution using an implementation-driven evaluation that leverages a dataset embedding nearly one million entries contributed by 550 crowdsensors over a year. Results show that *BeTogether* increases the quality of the collected data while reducing the overall resource cost compared to the cloud-centric approach.**

***Index Terms*—Crowdsensing, Mobile Sensing, Context Awareness, Environment Monitoring, Neighbor Discovery**

I. Introduction

Mobile crowdsensing, also known as *Mobile Phone Sensing* (MPS) or *crowdsensing* for short, holds the promise of monitoring physical phenomena at an unprecedented scale. Indeed, people may now gather various observations about the physical world using the sensors connected to their smartphones, along their journeys. This creates the potential to provide fine-grained observations that cover large areas over $24/7$ time periods, provided the engagement of a sufficiently large and diverse crowd [1], [2]. Still, the expected engagement of the crowd to sensing tasks differs depending on how the smartphones' owners contribute observations [3], [4], which may be either: (i) pro-actively, *aka participatory crowdsensing* [5], [6]; or (ii) passively in the background, *aka opportunistic crowdsensing* [7], [8]. The pros and cons of each approach are rather straightforward to infer.

Participatory crowdsensing has the potential to bring observations of higher quality since the end-user is conscious of carrying out a sensing task. Furthermore, supporting incentive mechanisms allow assigning the tasks to the most effective crowd [9], and complementary path planning schemes allow maximizing the number of tasks that each participant may achieve [10]. However, the participatory approach challenges the acquisition of fine grained knowledge across large areas and over time, due to the practical problem of "hiring" motivated volunteers. In particular, the massive collection of data often necessitates the participants' long term commitments, which is very hard to achieve in practice.

On the other hand, *opportunistic crowdsensing* makes it easy to collect a massive amount of observations across time and space. Indeed, there is no burden put on the end-users who simply need to have the mobile application installed on their smartphones. However, the challenge in this case is to ensure that the crowdsensors will contribute with sufficiently accurate knowledge. Since the participating users collect observations passively and without having to be conscious about it, the accuracy of the provided data with respect to the physical phenomenon under observation is uncertain. For instance, our analysis of the noise observations collected from the opportunistic crowdsensing application Ambiciti (formerly SoundCity) showed that less than 5% of the collected data were of sufficient quality to be considered for use in the mapping of urban noise pollution [11], [12]. Hence, opportunistic crowdsensing shifts the burden of quality sensing from the end-user to the crowdsensing system. In addition, fostering the engagement of large crowds –even if the engagement is passive and thus not demanding– requires the crowdsensing system to not deplete the smartphone resources. Supporting opportunistic crowdsensing at scale while meeting these requirements is the focus of this paper, which introduces the *BeTogether* middleware for collaborative crowdsensing at the edge (Figure 1).

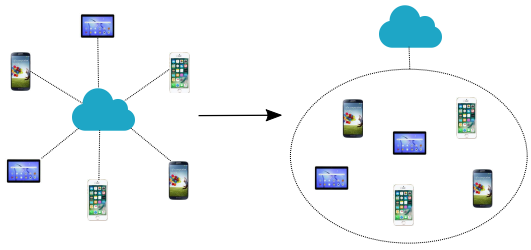

Fig. 1. From cloud-centric to collaborative crowdsensing at the edge

The design rationale of *BeTogether* directly derives from the location- and time-dependence of the crowdsensing of physical phenomena: co-located crowdsensors contribute related observations [13], and may thus collaborate toward improving the provided contributions and avoiding unnecessary duplication

978-1-7281-4657-7/20/$31.00 ©2020 IEEE

of work. More precisely, the crowdsensing tasks that need to be performed at the edge encompass: environment sensing, location provisioning, data processing, and uploading to the cloud. Then, while the replication of physical sensing within a group potentially allows for the gathering of more accurate knowledge, any of the other tasks only need to be performed by the most cost-effective group member(s). The fact that people tend to group [14], [15] as part of their daily activity, follow a daily/weekly routine [16], [17], and are most of the time still [11] further supports such a edge-based collaborative strategy to opportunistic crowdsensing. Still, a major challenge for the *BeTogether* middleware is to cope with the dynamics and heterogeneity of the crowdsensors [11]: the user activity, the resources available on the device, and the position of the sensors/smartphones are all criteria that characterize the crowdsensor's contribution to the upper layer application. We qualify as "*context*" the combination of these criteria. The *BeTogether* middleware then creates dynamic context-aware collaborative groups of crowdsensors, which are such that the peers within any given group: (i) Stay together for a long-enough time period so as to prevent constant changes and unnecessary grouping reconfiguration; (ii) Operate within the same physical environment (e.g., in-door vs out-door) and hence sense related physical phenomena; and (iii) Perform the same activity so that they behave alike –e.g., it is preferable that all the crowdsensors that collaborate either move together or are still. Then, upon the creation of a group, the middleware distributes the crowdsensing tasks to the most adequate group members according to the nodes' abilities –e.g., a smartphone located in the pocket cannot adequately sense the surrounding sound level. Overall, the *BeTogether* middleware fosters opportunistic crowdsensing that puts minimum burden on the end-users, while increasing the accuracy and resource-efficiency of the crowdsensing system.

After positioning the *BeTogether* solution with respect to the related work in the next section, the paper details the following contributions to the field of crowdsensing middleware:

- Introducing opportunistic, context-aware crowdsensing groups, so that groups deliver more accurate knowledge and enhance –both local and global– resource-efficiency compared to the gathering of the individual contributions at the cloud (§ III). Such grouping relies upon: (i) Node-specific context inference using online machine learning that accounts for the specifics of the device and user behavior; (ii) A set of utility functions that drive the distribution of the crowdsensing tasks among the group members so that any crowdsensing group does meet the objectives of improved quality of information and resource-efficiency; and (iii) An algorithm for the context-aware discovery of co-located crowdsensors and further task assignment.
- A prototype implementation of the *BeTogether* middleware for the Android platform, which paves the way for experimental evaluation but also the implementation of crowdsensing applications by third parties (§ IV).
- Evaluation of the *BeTogether* solution, which leverages a one-year data set of nearly one million entries from the Ambiciti crowdsensing application for noise pollution monitoring (http://ambiciti.io). Results show that collaborative crowdsensing at the edge would improve data quality, while its behavior adapts according to the context of the crowdsensors for the sake of resource-efficiency (§ V).

II. Background

The study of middleware supporting crowdsensing applications has been on the research agenda since the widespread adoption of smartphones that keep embedding an increasing number of sensors of increasing accuracy. Still, guaranteeing a sufficient level of quality for the contributed observations at a cost that both the producers and consumers of the data deem adequate is a lasting research challenge. The challenge is likely to last and the question is rather about a proper trade-off from the perspective of the involved stakeholders. Without being exhaustive, we review below the representative background for our work regarding quality- and/or resource-effective crowdsensing.

Participatory sensing: A large majority of the research on crowdsensing focuses on participatory systems and attempts to minimize the number of mobile devices that need to be recruited (to sense). The sensing coverage usually constitutes the main constraint used to formulate the related optimization problem. The goal henceforth consists in finding (the minimal) number of participants to cover a geographical area of interest and dispatch the participants accordingly. Several coverage-based recruitment strategies have been proposed, among which enhancing the data quality by introducing an incentive mechanism for crowdsensors to provide highly accurate and reliable data [9]. Further, multi-objective optimization serves maximizing (the weighted sum of): the average coverage, the usefulness of the data (with regards to redundancy), and the average number of active sensors, subject to the data quota constraint [18]. Still, participatory crowdsensing relies on the active engagement of users, which limits the achievable scale as well as may incur significant labor cost.

Opportunistic crowdsensing: Opportunistic crowdsensing comes with the promise of gathering observations at a massive scale at a potentially low cost by not requiring the active user participation beyond uploading the supporting app. Most solutions in that space are cloud- or fog-centric: data flows directly from each individual crowdsensor to a remote server/service that optimizes the data collection [19], [20] and analyzes the data [21]. Some approaches address resource-efficiency by taking into account the entire path that a mobile user is about to follow (e.g., leveraging data from the navigation app) [22], [19] so that a probabilistic mobility model allows estimating the redundancy of the provided observations. The crowdsensing platform may then select the minimal number of mobile crowdsensors that is necessary to cover a target area with a great probability, thereby minimizing the overall resource and energy consumption related to the sensing.

Another approach to resource-efficiency consists in discovering the nearby crowdsensors within the given Wi-Fi network so that a single node is tasked with sending local observations to the cloud after collecting them from its neighbors. In practice, the crowdsensor with the highest remaining level of energy, is periodically selected [23]. Unfortunately, such a solution requires every crowdsensor to be connected to a Wi-Fi network, which limits the applicability of the mobile crowdsensing. An alternative is then to leverage transient and D2D communication to discover the nearby devices. A proxy that will upload the data collected may then be selected within the D2D network [24]. In addition to the physical proximity (based on the communication range), other parameters (i.e., the number of neighbors, the remaining battery, the link quality, the stability of the link) have also been used in [25], [26] to determine which crowdsensor should operate as a proxy.

The aforementioned works mostly concentrate on selecting a crowdsensor to act as the network sink with the cloud, while solutions differ with respect to their selection criteria. In our work, we go one step further by considering the context-aware distribution of the various crowdsensing tasks among a set of co-located nodes so as to enhance the resource-efficiency and the data quality, both locally and globally. Indeed, by grouping together co-located crowdsensors that behave alike and observe compatible physical phenomena, they may collaborate at the edge so that there is no unnecessary duplication of tasks and more accurate data is sent over to the cloud. Still, the challenge is to guarantee that the group management is achieved in a way that does enhance resource efficiency, from the devices to the cloud, and the quality of the data gathered at the cloud, despite the heterogeneity of the crowdsensors and their inherent mobility. This is addressing such a requirement that led to the development of the *BeTogether* solution detailed in the next sections.

Opportunistic networking: The opportunistic networking of crowdsensors within a group requires special care. The opportunistic grouping of nodes has in particular been extensively studied in the early 2000s as part of the emergence of MANET (e.g., see [27]). However, the work was primarily focused on establishing a network on the fly, while latest networking technologies now directly support it. Indeed, D2D (Device-to-Device) communication, which is growing in popularity, offers a convenient base ground to opportunistically form a transient network, discover the nearby devices along with the services they can offer, and exchange data through the established D2D connections [28]. More recently, the variety of low-level D2D technologies (including Bluetooth Low Power, Wi-Fi Direct and the very recent Wi-Fi Aware) have been integrated and used jointly to support multi-technology networking [29], [30]. Building on this trend, we leverage innovation in D2D networking for the creation of the opportunistic crowdsensing groups. The *BeTogether* design specifically builds upon Wi-Fi Direct [31], [32], although alternative D2D communication technologies could be considered. Then, we customize the creation of opportunistic networks according to the specifics of opportunistic crowdsensing, that is, the networks are created according to the context, spanning: user behavior, physical environment, device attributes and the expected lifespan of the given context.

III. Context-aware Collaboration at the Edge

Our strategy to collaborative crowdsensing revolves around the opportunistic creation of groups of co-located crowdsensors, such that any given group fills the mandatory crowdsensing tasks: *Location*, *Internet connection*, *Sensors* for the target observations, and *Data processing* (over observations). The objective is further to ensure that the collaboration results in reducing the resource consumption due to crowdsensing, both locally and globally, compared to the cloud-centric solution. In particular, we leverage the relatively cheap cost –especially in terms of energy– of D2D communication compared to the costs associated with location management and cellular Internet connection to the cloud.

A. Context awareness

The *context* of a crowdsensor serves assessing the relevance of its peering with nodes in the D2D communication range for the sake of enhanced efficiency, that is, the ability to improve the gathered knowledge prior to transferring it to the cloud in a way that reduces resource-consumption.

1) Context metadata: The context characterizes the crowdsensor's collaboration profile, which subdivides into:

- **UA** (*User Activity*) refers to the mobility (or non-mobility) of the end-user, which the activity recognition module of the operating system implements. The module leverages machine learning [33] and relies on the sensors embedded in the smart device so as to determine if the user is still, walking, cycling or in a vehicle, etc..
- **PE** (*Physical Environment*) defines the position of the embedded sensors, which influences the given observations of physical phenomena. It detects if the crowdsensor is: in/out-pocket, in/out-door and under/above-ground. We leverage our previous work using online machine learning for such a user-centric inference, which learns the user behavior incrementally on the device [34], [35].
- **DA** (*Device Attributes*) characterizes the ability of the device to contribute to the various crowdsensing tasks. The attributes include the available networking and computing capabilities (i.e., type of Internet access –e.g., Wi-Fi–, (upload) bandwidth, remaining battery, CPU frequency, and memory size), the type of embedded sensors (e.g., {"Temperature", "Light", "Pressure", "Humidity", "Sound level"}) together with the related power consumption and accuracy.

2) Context duration: The effectiveness of the collaboration strategy depends not solely on the ability of grouping together crowdsensors that have compatible context [36], [37], but also on creating groups that last long enough. Indeed, the latter is critical to ensure that the benefit of the collaborative crowdsensing at the edge outperforms the overhead due to the group management/configuration.

We leverage online machine learning [34] in a way similar to the inference of *PE*, to predict how long the current *UA* and *PE* are going to last. That is, the predictor is continuously updated on the end device so that it keeps evolving according to the specifics of the user. Each time the user changes *UA*, the following new data instance is provided as input to the learning model: $\{t_{prev\_UA},\ UA_{prev},\ t_{cur\_UA} - t_{prev\_UA}\}$, where: t_{prev_UA} is the starting time of the previous *UA*, UA_{prev} is the previous value for *UA*, and t_{cur_UA} is the time at which the current *UA* started. The current time is defined as the tuple $\{$*day of week, hour of day, minute of hour*$\}$ because people tend to have repeated behavior at the week level. The same applies to the update of *PE* duration, with *PE* substituting to *UA* in the above instance definition.

Thanks to the online learning model, we predict the duration of the current *UA* (resp. *PE*) according to the current time and *UA* (resp. *PE*). The prediction of the *UA/PE* duration is a regression problem, rather than a classification. Many online learning algorithms may address the prediction model but fewer deal with regression. We have investigated three eligible algorithms: IBk (Instance Based k-nearest neighbor algorithm), KStar (an instance-based learner), and LWL (Locally Weighted Learning) [38]. We selected *LWL* as training algorithm because it provides the lowest RMSE and latency.

TABLE I
NEIGHBOR MATCHING RULES

Contexts	Matching Rules (i, j)
PE (except pocket) *PE duration*	**Consistent** for crowdsensors i and j **Greater than** D_{min} for both i and j
UA *UA duration*	**Consistent** for crowdsensors i and j **Greater than** D_{min} for both i and j
$RSSI$ value *Bearing (of moving)*	**Greater than** $RSSI_{min}$ for both i and j **Consistent** for crowdsensors i and j

3) Neighbors with matching context: In practice, the Received Signal Strength Indicator (RSSI) is used to estimate the distance between two crowdsensors (i.e., transmitter and receiver) [39], [13] and determine whether the two crowdsensors are close enough, i.e., $RSSI > RSSI_{min}$, with respect to the problem at hand [40]. We too consider the RSSI-based distance for assessing the relevance of grouping crowdsensors, which we enrich with the closeness of their respective context that should last for long enough (as predicted using machine learning). Bearing is also a parameter obtained from the operating system. Table I lists the resulting rules that determine the grouping of crowdsensors where the minimum duration is set according to the specific application. Given the set of the crowdsensors in the D2D communication range of smartphone i, we define the *neighbors* N_i of i as the nearby crowdsensors that matches all the rules. The grouping and the assignment of crowdsensing tasks to a set of neighbors then depend on their respective utilities to carry out the tasks.

B. Assessing the crowdsensor utilities

$$f(x, k, x_0) = \frac{1}{1 + e^{k(x_0 - x)}}$$

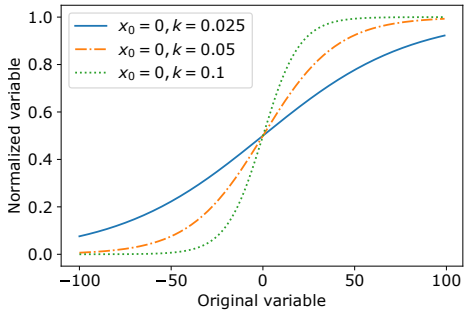

Fig. 2. Squashing function for normalization

We introduce a set of utility functions to evaluate the extent to which crowdsensors are eligible to carry out the various crowdsensing tasks. We rely on a classical squashing function for the normalization of the values used in the computation and comparison of the utilities. The squashing function f corresponds to a shifted and scaled logistic function (see Figure 2):

where: the range of f lays into $(0, 1)$, k is the logistic growth rate or steepness of the curve, and x_0 is the x-value of the midpoint of sigmoid, while x is the variable to be normalized. The values of k and x_0 are set according to the domain of the specific x. In the following, we denote k_f and x_f the values of k and x_0 for a given function f, while we define the actual values used in our prototype for the various parameters in Section IV.

The following tasks must be implemented within any collaborative group: *Coordinator* (implementing the D2D network access point for the group and assigning the crowdsensing tasks to the connected crowdsensors, i.e., *neighbors*), *Location provider* (supplying geographical coordinates), *Internet proxy* (providing Internet access and thus transferring data to the cloud), *Data aggregator* (analyzing together the collected data locally before sending to the cloud so as to, e.g., calibrate the sensors [41], analyze the data [12]), and, of course, *Sensors*.

1) Coordinator: The selection of the coordinator is based on the following criteria:

- *The number of neighbors* has a significant impact on the overall performance because increasing the number of collaborators enables analyzing more data locally and reducing the transfer of data to the cloud. In contrast, there is no benefit in creating groups with too few crowdsensors for which the minimum value δ depends on the application. Given a crowdsensor i and the set N_i of its neighbors, we define:

$$\Delta_i = f(\max\{0, |N_i| - \delta\}, k_\Delta, x_\Delta)$$

- *The occurrences of collaboration* between a crowdsensor i and its neighbors is another important parameter:

$$h_i = f(\frac{\sum_{j \in N_i} t_{collab}(i, j)}{|N_i|}, k_h, x_h)$$

where: $t_{collab}(i,j)$ is a numeric parameter indicating the number of times i and j have been collaborating, as defined in the history of i.

- *The UA and PE (except pocket)* of the crowdsensor should last sufficiently long. Given the predicted $duration(UA)$ (resp. $duration(PE)$) of the current *UA* (resp. *PE*) for i, we define:

$$d_i = f(\min\{duration(UA), duration(PE)\}, k_d, x_d)$$

- The *remaining battery capacity* of the crowdsensor should obviously be taken into account. Its normalized value is denoted as:

$$b_i = f(bat_i, k_b, x_b)$$

Next, provided the weights w_Δ, w_h, w_d, w_b, all $\in [0,1]$, set for the above criteria, we define the utility $u_c(i) \in (0, w_\Delta + w_h + w_d + w_b)$ of the crowdsensor i associated with acting as coordinator as the weighted sum of the above functions:

$$u_c(i) = w_\Delta.\Delta_i + w_h.h_i + w_d.d_i + w_b.b_i$$

2) Location provider: While the GPS location brings higher accuracy out-door, it also comes with higher energy consumption and latency, compared to the network-based location. This leads us to introduce the following normalized value l_i to the support of the positioning by i:

$$l_i = \begin{cases} f(acc_l, k_l, x_l) - f(pow_{gps}, k_{lp}, x_{lp}) & , \text{ GPS} \\ f(acc_l, k_l, x_l) - f(pow_{net}, k_{lp}, x_{lp}) & , \text{ Network} \\ -\infty & , \text{ Null} \end{cases}$$

where: we assume that any i advertises a single location service among *(GPS, Network, Null)*, acc_l is the accuracy of the advertised location service, and pow_{gps} (resp. pow_{net}) is the power consumed by the GPS (resp. network) location service. The utility function for crowdsensor i of being a location provider is thus defined as $u_l(i) \in (-\infty, 1 + w_d + w_b)$, which accounts for the location service source, location accuracy and remaining battery capacity:

$$u_l(i) = l_i + w_d.d_i + w_b.b_i$$

3) Internet proxy: The Internet proxy transfers the (analyzed) data provided by the group of crowdsensors to the cloud server. The service may be provided using either long range cellular or short range Wi-Fi transmission, while we assume that a node supporting both networks will offer the Wi-Fi based transmission by default. We define:

$$n_i = \begin{cases} f(bw_{up}, k_n, x_n) - f(pow_{wifi}, k_{np}, x_{np}) & , \text{ Wi-Fi} \\ f(bw_{up}, k_n, x_n) - f(pow_{cell}, k_{np}, x_{np}) & , \text{ Cellular} \\ -\infty & , \text{ Null} \end{cases}$$

where: bw_{up} is the upstream bandwidth for the internet interface; pow_{wifi} (resp. pow_{cell}) is the power consumption of Wi-Fi (resp. cellular Internet) transmission. The utility function for crowdsensor i acting as Internet proxy is then defined as $u_p(i) \in (-\infty, 1 + w_d + w_b)$, and it accounts for the Internet connection interface, up-link network bandwidth, and remaining battery capacity:

$$u_p(i) = n_i + w_d.d_i + w_b.b_i$$

4) Data aggregator: The data aggregator takes in charge the analysis of the sensing data collected locally. While lightweight data processing can be performed by the coordinator or by the proxy, complex data analysis may be delegated to a dedicated device that holds the necessary processing capabilities (spanning available memory, CPU frequency and remaining battery capacity). This results in the following definition for the supporting utility function $u_a(i) \in (-w_c, w_c + w_r + w_d + w_b)$:

$$u_a(i) = w_c.[f(cpu_i, k_c, x_c) - f(pow_c, k_{cp}, x_{cp})] + w_r.f(mem_i, k_r, x_r) + w_d.d_i + w_b.b_i$$

where: cpu_i is the CPU frequency, pow_c is the power consumption of the CPU, mem_i is the available memory, and coefficients w_c, w_r are the weights set for the metrics.

5) Sensors: The utility of a crowdsensor to carry out the sensing task depends on the accuracy of the contributed observations and their power consumption. In particular, a crowdsensor within a pocket/bag is ignored, which we filter out using the duration of the crowdsensor being out of the pocket (as defined by the PE_{out} value of *PE*):

$$d'_i = f(duration(PE_{out}), k_d, x_d)$$

which leads to the following utility function, $u_s(i) \in (-1, 1 + w_d)$:

$$u_s(i) = d'_i.[f(acc^s_i, k_a, x_a) - f(pow^s_i, k_{sp}, x_{sp}) + w_d.d_i]$$

where: acc^s_i (resp. pow^s_i) is the accuracy (resp. power consumption) of the sensor of type s on crowdsensor i.

6) Group utility: Given the above set of utility functions, the utility of a group with coordinator i is defined as:

$$U_G(i) = u_c(i) + u_l(l) + u_p(p) + u_a(a) + \sum_{s \in S} \sum_{n \in NS_i} u_s(n)$$

where:

- $l \in N_i$ is the node with the highest utility to act as the location provider among i's neighbors.
- $p \in N_i$ is the node with the highest utility to act as the Internet proxy among i's neighbors.
- $a \in N_i$ is the node with the highest utility to act as the data aggregator among i's neighbors.
- S is the set of sensor types requested by the upper layer crowdsensing application and the set $n \in NS_i$ defines the nodes with the highest utility to act as sensors s within N_i.

C. Grouping algorithm

Provided the utility functions, the grouping algorithm leverages the communication and discovery protocols implemented at the link layer by state of the art D2D communication technologies. Without loss of generality, our grouping algorithm

builds upon the Wi-Fi Direct technology [31]. In a nutshell, Wi-Fi Direct establishes a D2D opportunistic network through the discovery of peer nodes followed by the election of the node acting as the network's access point according to the criteria provided by the upper layer (i.e., in our case, the coordinator utility value provided by the *BeTogether* middleware).

Following, the grouping algorithm runs on every node *i*, either on-demand or on a periodic basis according to the network's dynamics. The algorithm decomposes into the following steps with the duration of each step being set by the upper layer application (see Algorithms 1 and 2 for detail):

1) Advertise presence with context metadata (UA_i, PE_i, DA_i) and their respective duration, so as to discover nearby peers and their context.
2) Compute and advertise provided utilities $u_c(i)$, $u_l(i)$, $u_p(i)$, $u_a(i)$, $u_{s_1}(i)$, ..., $u_{s_{|S|}}(i)$, together with receive the utilities from neighbors following their advertisement by peers.
3) Self-elect as coordinator if: $\forall j \in N_i : u_c(i) \geq u_c(j)$ and $\forall k \in N_i : (u_c(i) = u_c(k)) \wedge (ID(i) > ID(k))$ with the node's ID being unique and assigned by the middleware.
4) Create the D2D network if self-elected as coordinator and advertise the network to the neighbors that ultimately join if they have not joined another D2D network since the beginning of the period.
5) Assign the crowdsensing tasks to the best suited group members, as defined by their respective utilities.

Algorithm 1 Group Configuration

Require: i: crowdsensor
Input: T: set of tasks that should be allocated
Input: c_i: context of crowdsensors i
Input: $u_t(i)$: utility that crowdsensor i implements task t
Local variable: N_i: neighbors of i sharing the same context
Local variable: C_i: set of contexts provided by neighbors
Local variable: U_i: set of utilities provided by neighbors

$N_i \leftarrow \varnothing$, $C_i \leftarrow \varnothing$, $U_i \leftarrow \varnothing$
Register service (*BeTogether*, c_i)
Nearby Services Discovery to get C_i and N_i
for each task $t \in T$ **do**
  Compute $u_t(i)$
  Register service (t, $u_t(i)$)
end for
Nearby Services Discovery to get U_i
if $u_c(i)$ is the highest value in U_i **then**
  Create a Wi-Fi Direct group as group owner
  for each task $t \in T$ **do**
    Find crowdsensor k with highest utility $u_t(k) \in U_i$
    Invite crowdsensor k to implement t
  end for
else
  Find crowdsensor k with highest utility $u_c(k) \in U_i$
  Accept invitation and join group of k
end if

Algorithm 2 Nearby Service Discovery

Require: i: crowdsensor
Input: T: set of tasks that should be implemented
Output: N_i: neighbors of i sharing same context
Output: C_i: set of contexts provided by the neighbors
Output: U_i: set of contexts provided by the neighbors

Discover Wi-Fi Direct Service

Upon reception of advertisement (*BeTogether*, c_j) from j
if Rules matched (c_j, c_i) **then**
  $N_i \leftarrow N_i \bigcup \{j\}$, $C_i \leftarrow C_i \bigcup \{c_j\}$
end if
return C_i and N_i

Upon reception of advertisement (t, $u_t(j)$) from j
if $j \in N_i$ **then**
  $U_i \leftarrow U_i \bigcup \{u_t(j)\}, \forall t \in T$
end if
return U_i

Once a group is created, the arrival/departure of nodes is detected at the link layer by the underlying D2D protocol. This enables two approaches to the reconfiguration of the group: (i) on a periodic basis, or (ii) on-demand upon the detection of the departure/arrival of a group member. The latter performs very well and induces almost no cost when very few topology changes occur and when the context evolves slowly. On the other hand, it may lead to a unnecessarily high traffic and constant reassignment in a highly dynamic environment. Keeping in mind that, in practice, users are most of the time still (and hence evolving in the same context), we adopt an on-demand approach (based on context change) by default. Nevertheless, the middleware switches to periodic reassignment when the context gets highly dynamic.

IV. Prototype Implementation

Figure 3 depicts the architecture of the *BeTogether* middleware prototype: the (mobile) crowdsensor components run on Android 6.0+ smartphone or tablet; the server part in the cloud archives the data that are collected using a no-SQL database provided by Cloud Firestore.

The *BeTogether* prototype is available at https://github.com/sensetogether/BeTogether. The middleware running on the crowdsensors features the following components:

- *Crowdsensing manager:* It controls the workflow of the collaborative crowdsensing according to Algorithm 1, which includes the orchestration of the service discovery, context inference, computation of the utility functions, communication management, and the assignment of tasks. When a task is assigned to the device, the manager further instantiates the corresponding functionality/(ies) and starts a separate thread to perform the related task.
- *Context inference module:* It extracts the crowdsensor context (*UA*, *PE* and *DE*). It infers the *UA* value using the Google Activity Recognition API (https://developers.

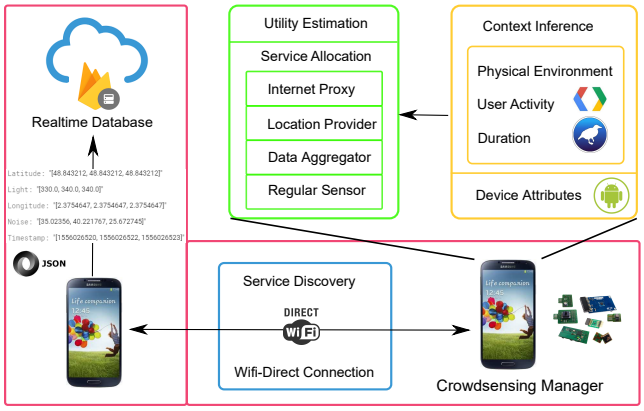

Fig. 3. *BeTogether* prototype architecture

google.com/location-context/activity-recognition). It also infers the *PE* values using the user-centric inference module available at https://github.com/sensetogether/ContextSense. The inference is performed using Hoeffding Tree which provides incremental decision tree induction (included in the Weka library at https://www.cs.waikato.ac.nz/ml/weka). It further predicts the time during which the *PE* and *UA* remain unchanged, using the LWL algorithm also supported by the Weka library.

- *Utility-based estimator and task assignment:* The inferred context is further used to estimate/compute the utility that assesses to which extent the crowdsensor should carry out a given task, i.e., coordinator, location provider, Internet proxy, data aggregator and environmental sensors.
- *Service discovery:* It uses *Wi-Fi Direct* [31], which provides both Device-to-Device communication and service discovery, at the data-link layer. Note that Wi-Fi Direct supports several service discovery protocols, such as Bonjour or UPnP, while our prototype uses the default Bonjour. A Wi-Fi Direct client plays the (logical) role of Access Point (AP) while other Wi-Fi Direct clients join the network that is governed by the AP. Such a network is equivalent to a classical Wi-Fi network.

TABLE II
PARAMETERS CONFIGURATION AND SCENARIO VARIABLES

Normalization Parameters	**Values**
k_Δ, x_Δ, k_h, x_h, k_d, x_d	1, 1, 1, 1, 0.1, $10mins$
k_l, x_l, k_{lp}, x_{lp}	0.1, $10m$, 0.1, $30mA$
k_n, x_n, k_{np}, x_{np}	10^{-5}, $10^5 kbps$, 0.01, $100mA$
k_b, x_b, k_{cp}, x_{cp}	10^{-3}, $10^3 mAh$, 0.01, $100mA$
k_c, x_c, k_r, x_r	10^{-3}, $10^3 MHz$, 10^{-3}, $10^3 MB$
k_a, x_a, k_{sp}, x_{sp}	0.1, 10%, 1, $0.1mA$
Collaboration Parameters	**Values**
w_Δ, w_h, w_d, w_b, w_c, w_r	1, 1, 1, 1, 1, 1

Finally, the *BeTogether* prototype implementation and testbed experiment detailed next, use the parameters defined in Table II. These parameters are set to smooth the normalization curve (Figure 2) considering nowadays' device hardware as a baseline. As for the weights w_x of the utilities, we consider an equality with no preference. The key parameters δ and D_{min} that determine the willingness to collaborate are application-dependent: (i) the minimum size of the group, as defined by δ, relates to the density of the contributing users, and (ii) the required stability of the group depends on the frequency of the measurements. We set both parameters according to the behavior of the Ambiciti application that supports the crowdsensing of urban noise measurements and from which we obtained the dataset of measurements gathered in Paris over the year 2017. Precisely, as the dataset is sparse, we set δ to 4, i.e., handling groups starting at 5 members, and noise measurements are by default taken every 5 minutes leading to set D_{min} to 5 minutes.

V. PERFORMANCE EVALUATION

We evaluate the performance of our approach, assessing: (i) the impact on the power consumption of the device running the *BeTogether* middleware in the background, and (ii) the potential benefit of the collaborative crowdsensing at the edge from the standpoint of data quality and communication cost based on a one-year dataset obtained from the Ambiciti application for urban pollution monitoring. We note that due to privacy and commercial concerns, the Ambiciti company shares the data with us in the framework of a collaboration agreement while data cannot be released openly.

A. Power consumption

TABLE III
ACTIVE POWER OF COMPONENTS FOR NEXUS 5X

Wi-Fi TX	Wi-Fi Scan	Cellular TX	Light Sensor	GPS RX
173 mA	2 mA	186 mA	0.2 mA	60 mA

We estimate theoretically the power consumption on a single crowdsensor; Table III shows the power of the main components used in our crowdsensing middleware. The reference values are for the Google Nexus 5X smartphone, as provided by the Android OS (https://source.android.com/devices/tech/power/values) according to the power profile provided by the manufacturer. Herein, we select the light sensor, which involves a power consumption comparable to that of most sensors [34], [13]. Note that the power consumption associated with the transmission depends on the transmission duration, which increases linearly with the packet size. We assume that cellular and Wi-Fi Direct communications have the same transmission speed.

Figure 4 shows the power consumption of a crowdsensor working individually at various sensing and upload frequencies. The upload is the most energy-demanding and the energy consumption can be reduced by lowering the upload frequency. As a comparison, Figure 5 provides the power consumption of the various nodes contributing to collaborative crowdsensing with regards to a high sensing frequency (every 1 minute) and upload frequency (every 10 minutes) for the individual case. Results show that the proxy always consumes the most energy due to the cellular transmission that takes place with the cloud,

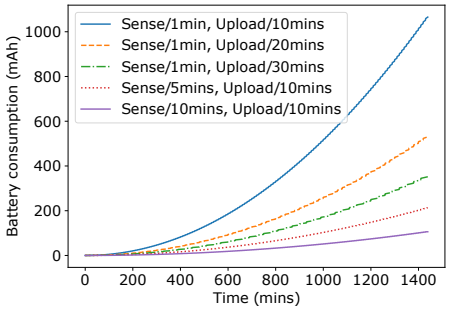

Fig. 4. Varying individual crowdsensing frequencies & Power consumption

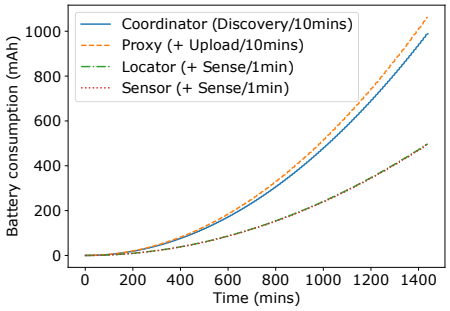

Fig. 5. Collaborative crowdsensing & Power consumption

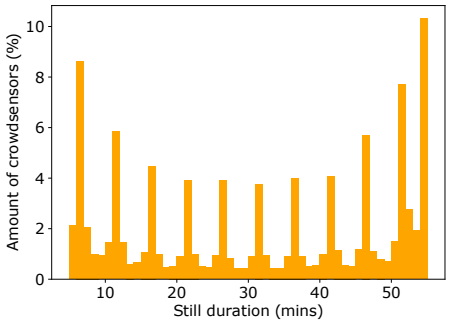

Fig. 6. Distribution of the still duration

followed by the coordinator that communicates with the nearby devices so as to distribute the tasks. Other group participants consume much less energy compared to the individual case, even-though this consumption includes the cost related to discovery and D2D transmission to the coordinator.

B. Dataset-driven evaluation

Dataset: We leverage a dataset produced by the Ambiciti (formerly called SoundCity) cloud-based crowdsensing application available on Google Play since 2015 and on Apple App-Store since 2016 (see http://ambiciti.io). Ambiciti monitors the noise pollution using the smartphone's microphone [42]. Our dataset contains 946,573 observations gathered both in-door and out-door in Paris over the year 2017 from 550 crowdsensors, where the average uploading duty cycle is around 5 minutes. Each observation provides: the uploading time-stamp, the location and (anonymized) ID of the contributing device, the noise level and its measurement bias, a description of the user activity (still, on foot, on bicycle, in vehicle, unknown), and whether the device is in/out-pocket (based on proximity). With 550 crowdsensors for the whole Paris city, the dataset is sparse. Hence, it does not provide the most suited case for opportunistic collaboration at the edge. Still, this allows us to assess the effectiveness of *BeTogether* even with a sparse deployment.

Analyzing the crowdsensor behavior: We first analyze the stability of the crowdsensors' context, as used for the configuration of groups, where we consider only the User Activity (*UA*). Indeed, the Physical Environment (*PE*) value is limited to the in-pocket case in the dataset, which influences only the sensor utility $u_s(i)$. The context is assessed per day. Starting with the initial location l of any crowdsensor i within the dataset, we consider that i changes group when it reaches another location l' that is more than the D2D range away from l, and repeatedly so with l' as the new reference location.

Figure 6 then shows the distribution of the duration of the crowdsensor staying within the above estimated group for all the crowdsensors of our dataset according to the device's location: it varies from 10 minutes to 60 minutes where we recall that we set $D_{min} = 5$ minutes as the minimum duration of the group. Hence, many crowdsensors remain at the same location long enough to group.

Figure 7 further compares the three following grouping strategies in terms of the average number of messages sent per crowdsensor per day so as to discover nearby crowdsensors: *periodic* (after every upload) that is the approach found in related work, *on-demand* (detected by Wi-Fi Direct), and *context-aware* that accounts for the crowdsensors' activities. In average, the amount of traffic generated by the on-demand strategy is 80.810% lower than the periodic approach, and the one of our context-aware grouping is 21.844% lower than the on-demand approach.

Analyzing the efficiency gain of grouping: In order to find the clusters of crowdsensors that are within D2D range (at 10m (re-scaled) away) from each other, we rely on the DBSCAN algorithm [43], which handles clusters that are arbitrarily shaped and that are of varying density. According to our parameter $\delta = 4$, that configures groups of size 5 and more, Figure 8 shows the distribution of the resulting group sizes in our dataset, with 79% of the groups being still.

Figure 9 further compares the average duration of the identified groups depending on whether the context is accounted for or not. Interestingly, results show that even in real life scenarios (including both still and mobile groups, not only considering still grouping as in Figure 6), our context-aware grouping finds groups of longer duration, which is up to 3.256 times of non context-aware grouping. As groups grow, the difference between a context-aware and non-context-aware approaches lowers because the likelihood of grouping co-located nodes having the same context gets higher. A decrease of lifetime is observed for groups of 12 members and more, which is partly due to the sparsity of our dataset and also the higher probability of members moving away from the group.

Although the crowdsensing data of our dataset is very sparse in time and space, the context-aware collaborative crowdsensing at the edge still brings benefit in terms of data quality and global resource consumption. It is especially efficient as the size of the group grows. Figure 10 shows that the *context-aware* collaborative approach delivers the best data quality: the collected measurement bias is reduced by up to 615% (resp. 407%) compared to an individual (resp. non-context-aware collaborative) crowdsensing approach. This is because of the selective sensing of *BeTogether* within each group, leading to the collection of the most accurate observations

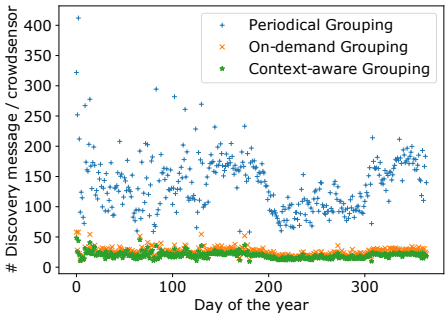

Fig. 7. Comparison of grouping strategies

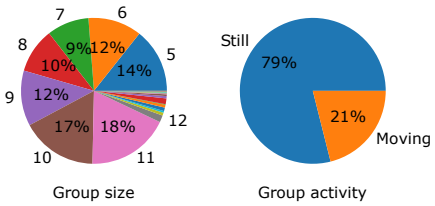

Fig. 8. Distribution of group properties

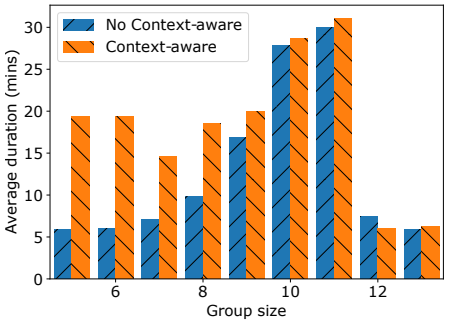

Fig. 9. Lifetimes of collaborative groups

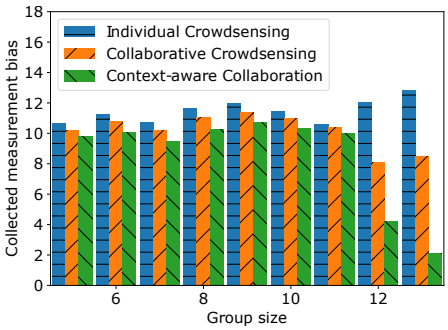

Fig. 10. Impact on collected data quality

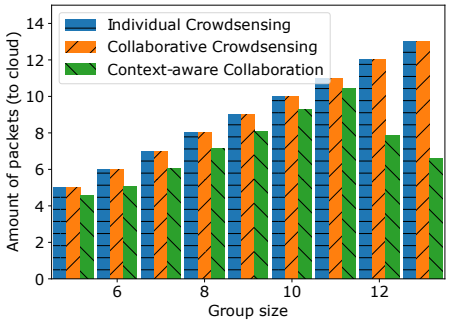

Fig. 11. Impact on uploaded data traffics

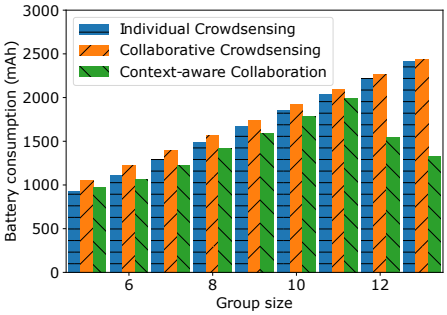

Fig. 12. Impact on overall battery consumption

rather than a simple average of the observations (non-context-aware collaborative crowdsensing) or than all the raw data (individual crowdsensing).

As shown in Figure 11, the amount of data that a collaborative and context-aware approach uploads to the cloud is reduced by up to 197% compared to both the individual and (non-context-aware) collaborative crowdsensing approach. There are two reasons for this: first, collaborative crowdsensing uploads only the aggregated data (i.e., concatenated hash tables) via the group proxy rather than uploading the raw data supplied by each crowdsensor (individual crowdsensing). Second, our context-aware collaboration also filters out the data that are of low quality and that are collected in-pocket, which reduces the amount of data aggregated and uploaded to the cloud.

Finally, Figure 12 focuses on the power consumed per hour, which is associated with both Wi-Fi Direct transmission (assuming that all the tasks are distributed) and cellular transmission: the power is reduced by up to 181% (resp. 183%) compared to individual (resp. non-context-aware collaborative) crowdsensing approach. Overall, these results show that the collaboration achieves a better data quality at a lower sensing and transmission cost when the group is larger because the tasks can be assigned to more devices, resulting in less task duplication and better context-aware filtering and aggregation.

VI. Conclusion

Opportunistic crowdsensing shows a great potential for monitoring the physical environment in a cost-effective and flexible way by empowering people to contribute as part of their daily life. While the growing participation of people helps covering urban-scale areas, it also necessitates to limit the increasing operational cost of the cloud-assisted infrastructure and to keep to a minimum the resource consumed by the hand-held devices. In this paper, we have introduced the *BeTogether* crowdsensing middleware, which implements a collaboration strategy at the edge so as to enhance the quality of the data transferred to the cloud while reducing the related communication cost and resource consumption. For this purpose, we have introduced a set of utility functions that assess to which extent a device should carry out a given crowdsensing task, while achieving a trade-off between the benefit for all (for the group) and the related cost for the device. We have also presented the prototype implementation of *BeTogether*, which builds upon the Wi-Fi Direct D2D communication technology for the creation and management of communication within a group. The evaluation of the *BeTogether* middleware solution using a large dataset from an existing crowdsensing application shows that our context-aware collaboration strategy improves the quality of the sensing data transferred to the cloud as well as reduce the resource consumption -both locally and globally. Our ongoing work now focuses on a distributed interpolation and aggregation approach running on the crowdsensors to achieve both higher sensing quality and efficiency.

Acknowledgment

The authors would like to thank: Ambiciti.io for granting us access to the specific data as part of research collaboration.

REFERENCES

[1] N. D. Lane, E. Miluzzo, H. Lu *et al.*, "A survey of mobile phone sensing," *Communications Magazine*, vol. 48, no. 9, 2010.

[2] J. Phuttharak and S. W. Loke, "A review of mobile crowdsourcing architectures and challenges: Toward crowd-empowered internet-of-things," *IEEE Access*, vol. 7, 2018.

[3] N. D. Lane, S. B. Eisenman, M. Musolesi *et al.*, "Urban sensing systems: opportunistic or participatory?" in *ACM International Workshop on Mobile Computing Systems and Applications*, 2008.

[4] R. K. Ganti, F. Ye, and H. Lei, "Mobile crowdsensing: Current state and future challenges," *Communications Magazine*, vol. 49, no. 11, 2011.

[5] B. Guo, Z. Yu, X. Zhou *et al.*, "From participatory sensing to mobile crowd sensing," in *IEEE International Conference on Pervasive Computing and Communication Workshops*, 2014.

[6] S. S. Kanhere, "Participatory sensing: Crowdsourcing data from mobile smartphones in urban spaces," in *International Conference on Distributed Computing and Internet Technology*. Springer, 2013.

[7] Y. Chon, N. D. Lane, F. Li *et al.*, "Automatically characterizing places with opportunistic crowdsensing using smartphones," in *ACM International Conference on Ubiquitous Computing*, 2012.

[8] H. Ma, D. Zhao, and P. Yuan, "Opportunities in mobile crowd sensing," *Communications Magazine*, vol. 52, no. 8, 2014.

[9] H. Jin, L. Su, H. Xiao *et al.*, "Incentive mechanism for privacy-aware data aggregation in mobile crowd sensing systems," *Transactions on Networking*, vol. 26, no. 5, 2018.

[10] M. Karaliopoulos, O. Telelis, and I. Koutsopoulos, "User recruitment for mobile crowdsensing over opportunistic networks," in *IEEE International Conference on Computer Communications*, 2015.

[11] V. Issarny, V. Mallet, K. Nguyen *et al.*, "Dos and don'ts in mobile phone sensing middleware: Learning from a large-scale experiment," in *ACM International Middleware Conference*, 2016.

[12] R. Ventura, V. Mallet, and V. Issarny, "Assimilation of mobile phone measurements for noise mapping of a neighborhood," *Journal of the Acoustical Society of America*, vol. 144, no. 3, 2018.

[13] C. Liu, J. Hua, and C. Julien, "Scents: Collaborative sensing in proximity iot networks," in *IEEE International Conference on Pervasive Computing and Communications Workshops*. IEEE, 2019.

[14] P. O. V. de Melo, A. C. Viana, M. Fiore *et al.*, "Recast: Telling apart social and random relationships in dynamic networks," *Performance Evaluation*, vol. 87, 2015.

[15] M. Saloni, C. Julien, A. L. Murphy *et al.*, "Lasso: A device-to-device group monitoring service for smart cities," in *IEEE International Smart Cities Conference*. IEEE, 2017.

[16] K. Farrahi and D. Gatica-Perez, "Discovering routines from large-scale human locations using probabilistic topic models," *Transactions on Intelligent Systems and Technology*, vol. 2, no. 1, 2011.

[17] T. M. Do and D. Gatica-Perez, "Human interaction discovery in smartphone proximity networks," *Personal and Ubiquitous Computing*, vol. 17, no. 3, 2013.

[18] Q. Zhu, M. Y. S. Uddin, N. Venkatasubramanian *et al.*, "Spatiotemporal scheduling for crowd augmented urban sensing," in *IEEE Internaltional Conference on Computer Communications*, 2018.

[19] S. Hachem, A. Pathak, and V. Issarny, "Probabilistic registration for large-scale mobile participatory sensing," in *IEEE International Conference on Pervasive Computing and Communications*, 2013.

[20] N. D. Lane, Y. Chon, L. Zhou *et al.*, "Piggyback crowdsensing (pcs): energy efficient crowdsourcing of mobile sensor data by exploiting smartphone app opportunities," in *ACM International Conference on Embedded Networked Sensor Systems*, 2013.

[21] P. P. Jayaraman, J. B. Gomes, H.-L. Nguyen *et al.*, "Scalable energy-efficient distributed data analytics for crowdsensing applications in mobile environments," *Transactions on Computational Social Systems*, vol. 2, no. 3, 2015.

[22] X. Sheng, J. Tang, and W. Zhang, "Energy-efficient collaborative sensing with mobile phones," in *IEEE International Conference on Computer Communications*, 2012.

[23] J. Dutta, P. Pramanick, and S. Roy, "Noisesense: Crowdsourced context aware sensing for real time noise pollution monitoring of the city," in *IEEE International Conference on Advanced Networks and Telecommunications Systems*, 2017.

[24] T. Higuchi, H. Yamaguchi, T. Higashino *et al.*, "A neighbor collaboration mechanism for mobile crowd sensing in opportunistic networks," in *IEEE International Conference on Communications*, 2014.

[25] V. F. Mota, D. F. Macedo, Y. Ghamri-Doudane *et al.*, "Managing the decision-making process for opportunistic mobile data offloading," in *IEEE Network Operations and Management Symposium*, 2014.

[26] V. F. Mota, T. H. Silva, D. F. Macedo *et al.*, "Towards scalable mobile crowdsensing through device-to-device communication," *Journal of Network and Computer Applications*, vol. 122, 2018.

[27] J. Liu, D. Sacchetti, F. Sailhan *et al.*, "Group management for mobile ad hoc networks: Design, implementation and experiment," in *ACM International Conference on Mobile Data Management*, 2005.

[28] J.-W. Yoo and K. H. Park, "A cooperative clustering protocol for energy saving of mobile devices with wlan and bluetooth interfaces," *Transactions on Mobile Computing*, vol. 10, no. 4, 2010.

[29] T. Kalbarczyk and C. Julien, "Omni: An application framework for seamless device-to-device interaction in the wild," in *ACM International Middleware Conference*, 2018.

[30] C. Julien, C. Liu, A. L. Murphy *et al.*, "Blend: practical continuous neighbor discovery for bluetooth low energy," in *ACM International Conference on Information Processing in Sensor Networks*. ACM, 2017.

[31] W.-F. Alliance, "Wi-fi peer-to-peer services (p2ps) technical specification," Version 1.2, Tech. Rep., 2015.

[32] K. Liu, W. Shen, B. Yin *et al.*, "Development of mobile ad-hoc networks over wi-fi direct with off-the-shelf android phones," in *IEEE International Conference on Communications*, 2016.

[33] M. Elhoushi, J. Georgy, A. Noureldin *et al.*, "A survey on approaches of motion mode recognition using sensors," *Transactions on Intelligent Transportation Systems*, vol. 18, no. 7, 2016.

[34] Y. Du, V. Issarny, and F. Sailhan, "User-centric context inference for mobile crowdsensing," in *ACM/IEEE International Conference on Internet of Things Design and Implementation*, 2019.

[35] ——, "When the power of the crowd meets the intelligence of the middleware: The mobile phone sensing case," *ACM SIGOPS Operating Systems Review*, vol. 53, no. 1, 2019.

[36] A. B. Abkenar, S. W. Loke, W. Rahayu *et al.*, "Energy considerations for continuous group activity recognition using mobile devices: The case of groupsense," in *IEEE International Conference on Advanced Information Networking and Applications*, 2016.

[37] A. B. Abkenar, S. W. Loke, A. Zaslavsky *et al.*, "Garsaaas: Group activity recognition and situation analysis as a service," *Journal of Internet Services and Applications*, vol. 10, no. 1, 2019.

[38] I. H. Witten, E. Frank, M. A. Hall *et al.*, *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2016.

[39] A. Bose and C. H. Foh, "A practical path loss model for indoor wifi positioning enhancement," in *IEEE International Conference on Information, Communications & Signal Processing*, 2007.

[40] T. Higuchi, H. Yamaguchi, and T. Higashino, "Context-supported local crowd mapping via collaborative sensing with mobile phones," *Pervasive and Mobile Computing*, vol. 13, 2014.

[41] F. Sailhan, V. Issarny, and O. Tavares-Nascimiento, "Opportunistic multi-party calibration for robust participatory sensing," in *IEEE International Conference on Mobile Ad Hoc and Sensor Systems*, 2017.

[42] R. Ventura, V. Mallet, V. Issarny *et al.*, "Evaluation and calibration of mobile phones for noise monitoring application," *Journal of the Acoustical Society of America*, vol. 142, no. 5, 2017.

[43] M. Ester, H.-P. Kriegel, J. Sander *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *ACM Conference on Knowledge Discovery and Data Mining*, 1996.