Hindawi Publishing Corporation
Mobile Information Systems
Volume 2016, Article ID 9681842, 26 pages
http://dx.doi.org/10.1155/2016/9681842

Review Article

A Survey on Smartphone-Based Crowdsensing Solutions

Willian Zamora, Carlos T. Calafate, Juan-Carlos Cano, and Pietro Manzoni

Department of Computer Engineering, Universitat Politècnica de València, Camino de Vera S/N, 46022 Valencia, Spain

Correspondence should be addressed to Willian Zamora; wilzame@posgrado.upv.es

Received 19 May 2016; Accepted 16 October 2016

Academic Editor: Dik Lun Lee

In recent years, the widespread adoption of mobile phones, combined with the ever-increasing number of sensors that smartphones are equipped with, greatly simplified the generalized adoption of crowdsensing solutions by reducing hardware requirements and costs to a minimum. These factors have led to an outstanding growth of crowdsensing proposals from both academia and industry. In this paper, we provide a survey of smartphone-based crowdsensing solutions that have emerged in the past few years, focusing on 64 works published in top-ranked journals and conferences. To properly analyze these previous works, we first define a reference framework based on how we classify the different proposals under study. The results of our survey evidence that there is still much heterogeneity in terms of technologies adopted and deployment approaches, although modular designs at both client and server elements seem to be dominant. Also, the preferred client platform is Android, while server platforms are typically web-based, and client-server communications mostly rely on XML or JSON over HTTP. The main detected pitfall concerns the performance evaluation of the different proposals, which typically fail to make a scalability analysis despite being critical issue when targeting very large communities of users.

1. Introduction

The use of mobile phones has experienced a significant increase in the past decade. In fact, according to the 2015 ITU World Telecommunications report [1] for 2015, the ratio of Cellular phone subscriptions was 97%, which represents 7084 million subscribers in the world. In addition, this subscriber increase is reflected in the technological advantages offered by mobile devices. Furthermore, mobile devices available nowadays have a high computational power and include different communication technologies (e.g., WiFi, 4G, and Bluetooth) and have multiple embedded sensors (GPS, gyroscope, accelerometer, microphone, and camera, among others). This technological growth, together with the increasing number of subscribers, has caused the community of researchers and developers to create different applications based on smartphones as sensors.

Pioneering research anticipated this arising of new applications, describing them as "participatory sensing" [2] or "people-centered sensing" [3]. In both cases, the idea is that the user should be able to gather data anywhere, anytime, by making use of mobile sensor devices for information retrieval, processing, and sharing. Later on, researchers considered this new paradigm as a subtype of crowdsensing denoted as "mobile phone sensing" [4].

Mobile phone sensing benefits from the processing and communication capabilities of available smartphones which, combined with one or more sensors, become an enabling technology to support different types of applications. Moreover, mobile crowdsensing relies on a large number of participants to collect data from the environment through its integrated sensors and, after capturing the data, these are sent to a server to perform data mining tasks including data fusion, analysis, and information dissemination. Typically, sensors that register participant information (e.g., location, movements) and environmental data (e.g., images, sounds) are very common. On top of that, some solutions use external sensors, which are integrated into the mobile solution through its communication interfaces, including sensors for environmental pollution and health monitoring. In this sense, mobile crowdsensing provides new perspectives for improving living conditions in our digital society. A general example of mobile crowdsensing solution is shown in Figure 1.

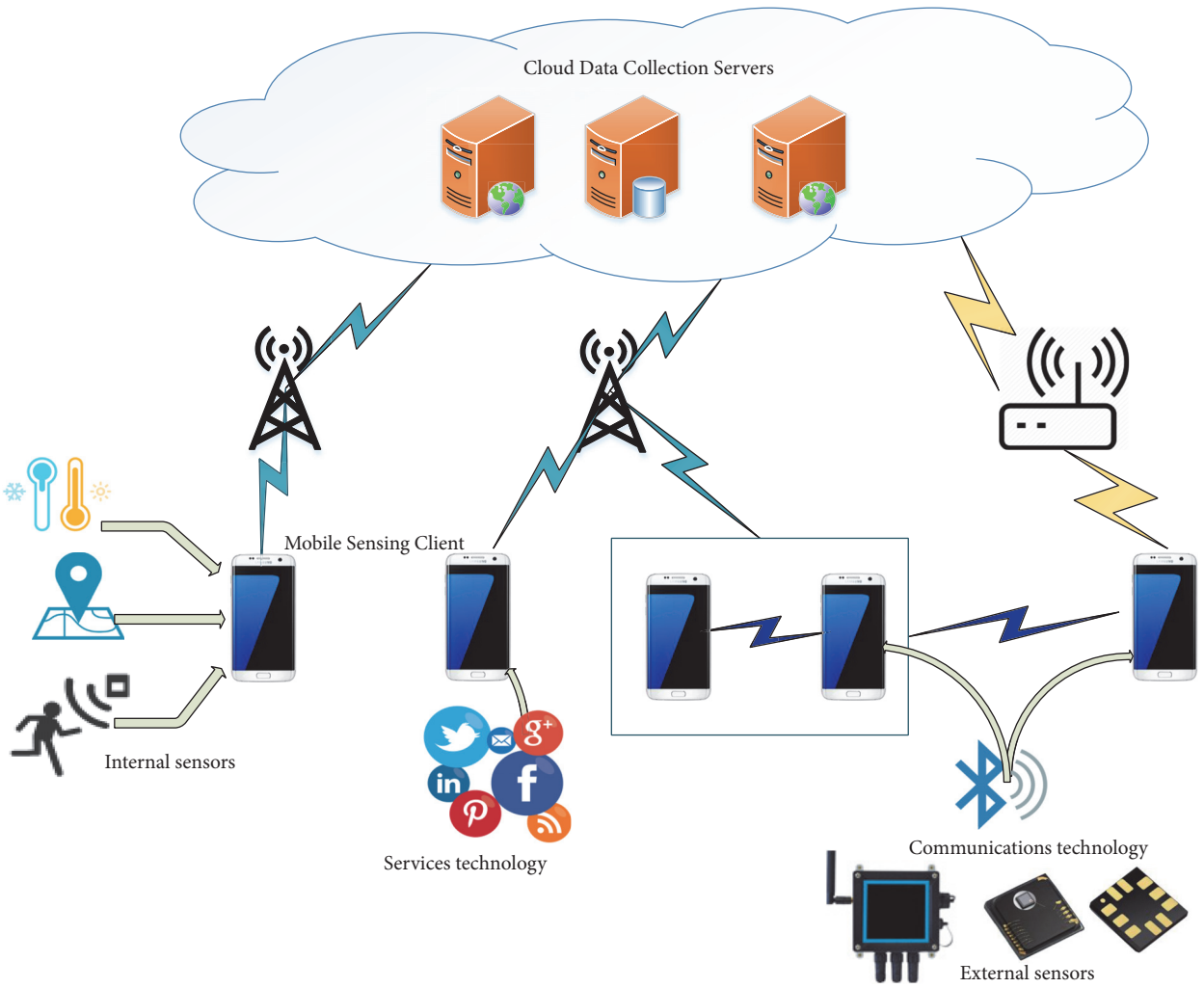

Figure 1: Generic structure of crowdsensing solutions.

Concerning mobile crowdsensing applications, Figure 2 shows that they have experienced a significant increase in the last 5 years. Specifically, researchers have focused their efforts on various areas including environment monitoring [5–8], transportation and urban sensing [9–13], healthcare [14–18], social issues [19–23], and others [24–28]. The different crowdsensing proposals available are characterized by having different designs and involve different architectural levels. For instance, some authors propose solutions they call framework, middleware, or system, among other terms. No matter which term is used, these solutions can have a global approach (full architecture) or only specify a subset of the architecture by describing one or more components.

The existence of a high number of proposals, and the absence (to date) of a survey that properly organizes such information, has led us to write this paper. In this work, we start by proposing a reference client-server architecture where the sensing device is the Mobile Sensing Client (MSC) and the server is the Cloud Data Collection Server (CDCS). Our idea is to propose an architecture that is generic and

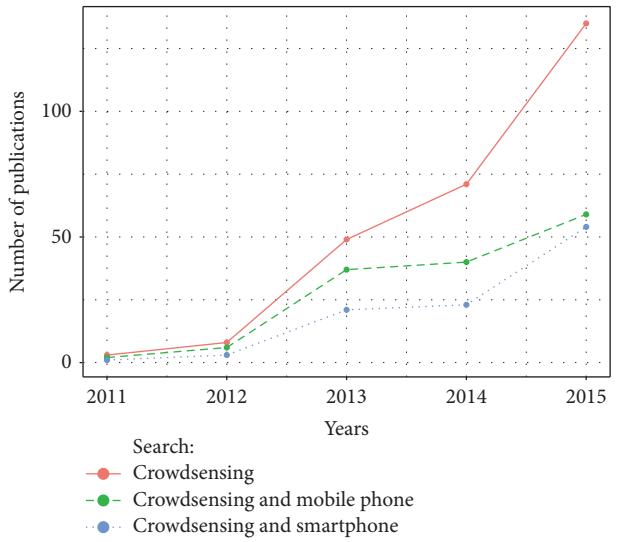

Figure 2: Number of crowdsensing-related proposals in the past 5 years.

flexible enough to accommodate any existing solution. With this in mind, we have identified whether the main contribution of each publication focuses on the client, on the server, on the transmission, or on some specific component. For client-side proposals, we determined whether their main contribution is in the processing or the data capture element, discriminating between sensor and data administration. For the server-side proposals, we determined the actual contributions in terms of mobile sensing tasks, dimensional analysis of data, and cloud services. Finally, we determined the contribution of those proposals focusing on data communications.

The paper is organized as follows: in the next section we present some related surveys on this topic. In Section 3 we provide an overview of the proposed architecture. Then, in Section 4, we make a detailed analysis of the proposed architecture. Section 4 provides the actual survey results, detailing the contributions made at the client and server sides, as well as to the end-to-end communications approach. Open research issues are then discussed in Section 5. Finally, in Section 6, we present our conclusions and future work.

2. Related Works

In recent years, the rise of solutions in the field of mobile crowdsensing is attracting huge interest because of the large amount of data that sensors can provide when relayed via mobile phones. Nonetheless, there are quite few surveys that actually study and summarize the many existing proposals. Below we proceed to describe briefly the different surveys found in the literature according to the chronological order of their publication.

Lane et al. [4] made a pioneer survey addressing the use of mobile phones as sensors, analyzing the significant progress mobile phones have experienced in order to incorporate multiple sensors. Also, they describe various existing proposals according to certain algorithms, applications, and systems developed to date. Similarly, they describe some proposals grouped by areas such as transportation, environmental monitoring, and health. They also propose an architecture composed of three different elements dedicated to sensing (mobile phone), learning (analysis), and informing (shared data).

Ganti et al. [29] proposed the term mobile crowdsensing (MCS) and described a reference architecture. This survey only showed a few proposals categorized as participatory (users are involved) and opportunistic (users are not involved). Additionally, it identifies some characteristics that influence these solutions such as limited resources, privacy and security, data integrity, data aggregation, and data analytics.

Khan et al. [30] present a taxonomy where they differentiate between personnel sensing, social sensing, and public sensing. This classification is performed from the point of view of participatory and opportunistic sensing.

Zhang et al. [31] propose an approach which characterizes the various crowdsensing proposals in four stages: task creation, task assignment, individual task execution, and crowd data integration. These features are described as what, when, where, who, and how (4W1H).

Table 1: Crowdsensing surveys.

Solutions	Year	# publications reviewed
Lane et al. [4]	2010	25
Ganti et al. [29]	2011	13
Khan et al. [30]	2013	43
Calabrese et al. [38]	2014	26
Zhang et al. [31]	2014	15
Zhang et al. [32]	2015	32
Jaimes et al. [33]	2015	22
Guo et al. [37]	2015	41

More recently, both Zhang et al. [32] and Jaimes et al. [33] proposed a classification based on incentive mechanisms for mobile crowdsensing. The former classified incentives into three categories: entertainment, services, and economic. The latter used incentive mechanisms as metrics to evaluate crowdsensing and introduced a tree-level taxonomy for crowdsensing incentive mechanisms. In the same context, different authors [34–36] propose incentive mechanisms that rely on auction techniques for evaluating quality awareness in the mobile crowdsensing context. In particular, the first uses combinatorial auction models, while the second extends that work by introducing more fine-grained techniques; concerning the third work, it proposes a framework that integrates incentives, data aggregation, and data perturbation mechanisms. However, they did not propose any reference architecture, addressing solely the taxonomy of their proposals and the algorithms supporting these proposals.

Guo et al. [37] propose a new sensing paradigm called Mobile Crowd Sensing and Computing (MCSC) that empowers ordinary citizens to contribute data sensed or generated from their mobile devices, aggregating and fusing the data in the cloud for crowd intelligence extraction and human-centric service delivery. This paper proposes a taxonomy and a reference architecture for MCSC. In the taxonomy the proposals are classified as mobile sensing (user involvement, data contribution, user awareness, and sampling), crowd data collection (networking, incentives, and scale), crowdsourced data processing and intelligence extraction (processing architecture, intelligence, purpose, data mining, and data quality), hybrid human-machine system, and security and privacy. With regard to the architecture, the proposals presented in this paper are divided into several levels: crowdsensing, data collection, data processing, and applications.

As it quickly becomes evident through this brief state-of-the-art analysis, to date only a few surveys specifically addressed existing crowdsensing solutions, being that some authors focused on specific issues such as incentives, and yet others focused on sensing styles. Our survey proposes a reference client-server architecture and then, based on that proposal, proceeds to classify up to 64 different proposals, thus providing a wider view than the surveys presented before on this topic (see Table 1 for details). Notice that the number of peer-reviewed publications only takes into account those references actually classified according to the proposed taxonomies.

3. Mobile Crowdsensing: Reference Architecture

In this section we propose a client-server design which can be adapted to the different mobile crowdsensing architectures available in the literature. By making the different proposals fit into our architecture, in sections that follow, it will then become straightforward to compare the different proposals in terms of scope, complexity, and completeness.

Our proposed architecture integrates two main modules: the Mobile Sensing Client (MSC) module and the Cloud Data Collection Server (CDCS) module. These two modules are connected to each other through a data transmission network, as shown in Figure 3. The MSC is the mobile phone or the set of mobile phones that provide sensing functionality by capturing data and then relaying that data to the CDCS. The latter is a single server or a server farm that allows receiving, processing, analyzing, and sharing sensed data. Typically, data sharing also includes the delivery of reports to participants (MSC).

For both the MSC and the CDCS we have considered four subcomponents, some of them sharing common characteristics on both MSC and CDCS. For instance, both Client and Server User Interfaces provide a graphical interface to a regular user or to the system administrator through the respective Interface Managers. On the bottom of the architecture, the Server and Client Communications Managers have also a similar purpose, typically being the component on the client that establishes connections with the server component since it should be always available. Nevertheless, configuration and task instructions, along with data reports, can also be transmitted from server to client through a push procedure.

The data management components at client and server also have some similarities, both being responsible for data processing, storage, and query. The main difference between these subcomponents is that, in the CDCS, the computation, storage, and analysis are made at a level and dimension that are clearly superior to the one made at the client, which has fewer resources.

Two distinctive components in our architecture are the Client Sensor Manager (CSM), responsible for the administration of the sensors, and the Server Task Manager (STM), which handles different tasks mostly related to data processing.

Below we proceed to describe the different architectural elements in more detail.

3.1. Mobile Sensing Client. In the scope of mobile crowdsensing, the main goal of the mobile client devices is performing data sensing and forwarding sensed data to the main server, although global data reports can also be returned to clients.

Concerning the target areas to be sensed, these can differ greatly depending on the type of application (inside buildings, outdoor, underground, in public places, etc.). In addition, each specific application will also have different requirements in terms of required sensors. For instance, sensors able to monitor the environment greatly differ from those able to monitor social interactions or the effectiveness of public transportation. In addition, sensing tasks can be triggered automatically (either periodically or based on events) or manually through an explicit user intervention. Typically, automatic mechanisms follow server instructions, while manual interactions are made possible through a User Interface specifically developed for that purpose. Independently of the actual mode of operation, the application can offer certain incentives in the form of a game [39] or another, to motivate users into adopting it. Such incentives become especially important when the user interest about the global generated data, which are based on the aggregation and processing of all measurements at the server, remain low (e.g., data being sensed is not a concern to the user); in those cases, complementary sources of motivation are required to make users run the crowdsensing application.

Focusing on the client architecture, Figure 4 shows that, to support all user activities, we have a set of managers responsible for all tasks: Client Interface Manager (CIM), Client Data Manager (CDM), Client Sensor Management (CSM), and Client Communications Manager (CCM). Each of these four components has a controller subcomponent, being the different controller elements, the ones actually responsible for supporting bidirectional interactions between the different system elements.

We now proceed to detail each of the client components in detail.

3.1.1. Client Interface Manager (CIM). This component allows applications to interact with the user (User Interface GUI). The User Interface allows displaying the values obtained from sensors in real time, to visualize previous traces through a query to its internal data storage or to query the server in order to retrieve global data reports about a certain target area. The values can be visualized through the use of graphics, maps, or other forms of representation. To achieve this goal two subcomponents are proposed: the Client User Interface and the Interface Controller.

(i) Client User Interface. It allows configuring the different parameters associated with sensing tasks, such as regulating the data acquisition frequency, defining when data should be sent to the server, and also when captures should start and stop, among others. It can also show the user feedback about ongoing or past captures, as well as global reports. It is worth highlighting that some crowdsensing solutions have no interface at the client side, meaning they only process captured data and relay them to the server.

(ii) Interface Controller. It provides the needed services to format data for presentation through the User Interface. For this endeavor, it must interact with the local storage or with the server, and it may rely on different external libraries as well (e.g., graphical representation of captured values in a map using Google Maps).

3.1.2. Client Data Manager (CDM). This element, responsible for data handling and storage, is one of the main architectural elements at the client. It is composed of five different subcomponents: data controller, Plugin Extensions, data processing,

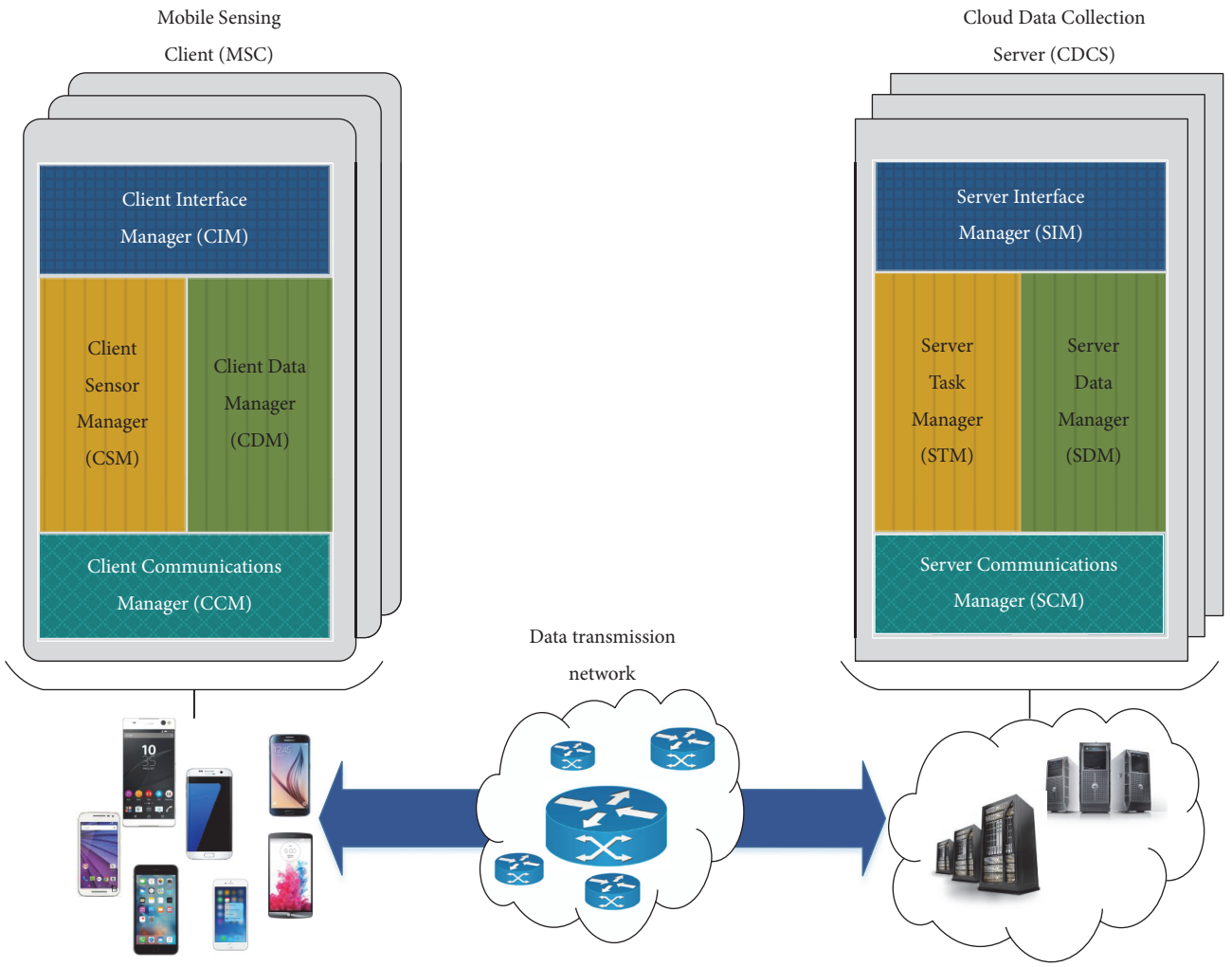

Figure 3: Proposed mobile crowdsensing architecture.

local storage, and query. We now proceed to detail each of them.

(i) Data Controller. It is the most critical subcomponent, providing the services and functions required to interact with the different subcomponents of the CDM. This interaction is made with the client via the Interface Controller, with the server via the Communications Controller and with the sensors via the Sensor Controller. In addition, it is able to handle data collection tasks as defined by the user or defined by the server through task pushing. It includes classes and methods to start, stop, and configure these tasks.

(ii) Plugin Extensions. This element allows integrating specialized plugins for a specific task such as data analytics or to add listeners to social networks like Facebook and Twitter, among others. The advantage of these plugins is that they can be easily incorporated into mobile devices via repositories such as Google Play or similar ones. Additionally, it allows plugging in a set of algorithms that perform functions including audio processing, online programming algorithms, and spatial coverage analysis.

(iii) Data Processing. This element processes raw data based on application requirements before displaying them to the end user or submitting them to the server. Although data processing can also be executed at the server side (CDCS), doing it at the client allows reducing the amount of unnecessary data produced by sensors, while also maximizing energy savings and communications bandwidth, and so it is often preferred. Typically, data processing elements include either filtering or aggregation or both functions. An example of filtering is the removal of unnecessary data fields. Examples of aggregation/fusion of data include the unification of data from different sensors or of different samples from a same sensor.

(iv) Local Storage. This element allows storing the captured data in a local data structure, which is usually a simple database like SQLite. Some solutions available in the literature skip this component, and they only process data and forward them to the CDCS. The local storage allows users to perform queries, inserts, updates, and deletes to the data according to application requirements. Typically, when storing data coming from sensors, it is often preprocessed before storage.

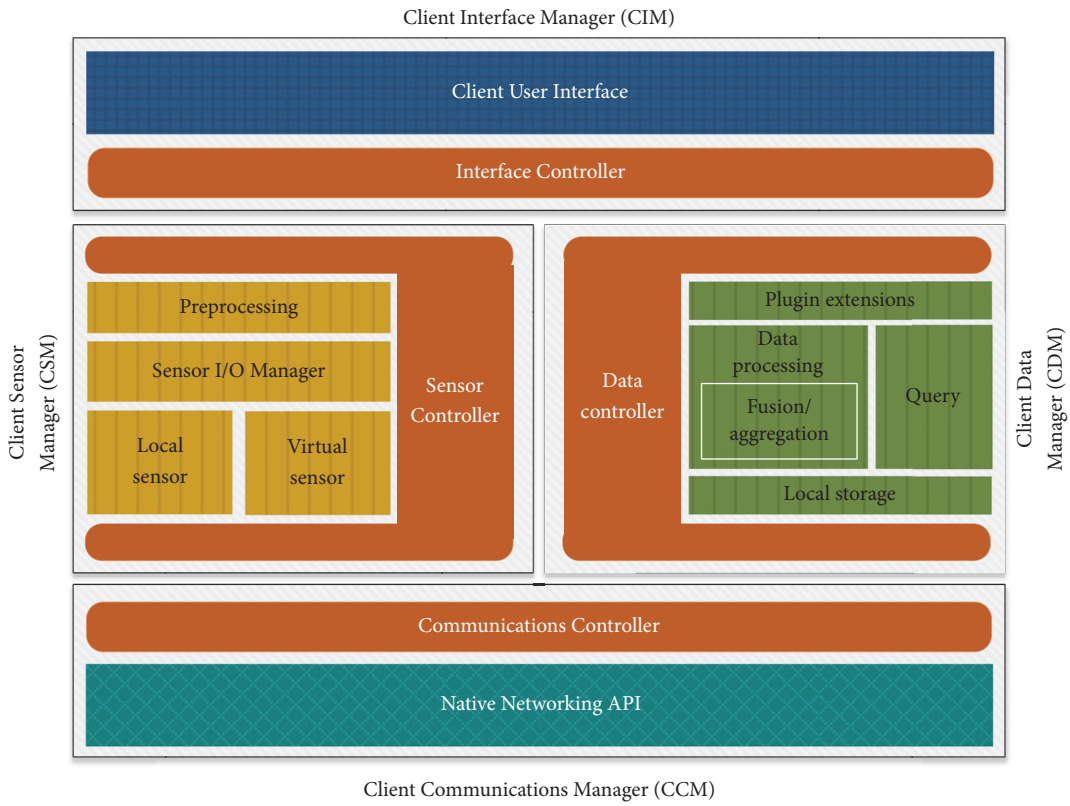

Figure 4: Mobile Sensing Client (MSC) components.

In the context of crowdsensing applications, the main types of data stored include location information, energy levels, and sensor-specific values.

(v) Query. This element allows, through structured language queries, accessing data from sensors. In particular, it will interact with all the components that make up the CDM. Among its typical features, one that stands out is the use of mobile analytics for optimizing data streaming from sensors. In some cases, this component facilitates the interaction with external databases at the CDCS in order to retrieve global data reports.

3.1.3. Client Sensor Manager (CSM). The Client Sensor Manager is the element responsible for the actual sensing tasks. Typically, it relies on high-level sensors abstractions to manage the underlying physical sensors (internal or external) as well as virtual sensors. Its functions usually include sensor discovery and sensing capabilities. Furthermore, it manages the sensor sampling frequency, as well as the preprocessing of captured data. The preprocessing executed at the CSM is only performed if necessary, and considering the actual characteristics of the sensor. Finally, the integration of external sensors and virtual sensors is performed by the Sensor Controller via the communications manager.

The CSM has five subcomponents: Sensor Controller, Preprocessing, Sensor I/O Manager, Physical Sensor, and virtual sensor. Below we describe each of its components.

(i) Sensor Controller. It enables access to the services offered by the Sensor Manager, thus providing access to virtual sensors, gyroscope, and GPS, among others.

(ii) Preprocessing. It allows the data delivered by the Sensor I/O Manager to be processed before being passed to other components. An application example is an audio capture which must be classified into voice and nonvoice regions, so that the individual speaker is segmented. Another example is the raw accelerometer data that is provided for the three axes, which can be combined to obtain the total value. In some cases these raw data can be processed at both CSM and CDM.

(iii) Sensor I/O Manager. It allows a level of abstraction for accessing both physical and virtual sensors, getting the raw data for subsequent treatment. This way, upper layers do not have to be aware of the type of sensor (physical/virtual) and its actual location.

(iv) Local Sensor. These are sensors available either on mobile devices themselves or external nearby sensors directly accessible by the mobile device. Concerning the type of sensor, most internal sensors used belong to the generic or media type. Generic sensors are those sensors embedded in mobile devices for general-purpose applications. Examples of these sensors include GPS, accelerometer, gyroscope, magnetometer, and barometer. With regard to media sensors, it refers to embedded sensors that provide support to multimedia

applications via microphone or camera. Finally, external sensors typically extend the sensing functionality by providing sensing capabilities not supported by the smartphone itself.

(v) Virtual Sensor. Virtual sensor is a logical type of sensor based on an abstract class that acts as a wrapper, encapsulating information that can be produced by a real sensor, a mobile phone, or a combination of other virtual sensors. Virtual sensors can have multiple input data streams that can be other virtual sensors or sensors accessible through a network, but there can be only one output data stream toward the sensing application. The GSN standard data model [40] is a good example of such a class of sensors.

3.1.4. Client Communications Manager (CCM). The Client Communications Manager is responsible for the transmission and reception of the data through the network. Since nowadays mobile phones include several communication interfaces including WiFi, Bluetooth, or Cellular, this empowers them to communicate in all sorts of environments, being able to adapt to different network topologies (centralized, distributed, or hybrid). The MSC may transfer the data to a primary server (centralized), toward several servers (distributed), or among themselves (peer-to-peer). The latter occurs when there are nodes that serve as intermediaries for the transmission of data between nodes and that have a limited ability to process and filter data from the sensor.

The CCM is composed of two subcomponents: the Communications Controller and the Native Networking API. In detail, these subcomponents are responsible for the following tasks.

(i) Communications Controller. It provides access to the services of the underlying communications network, allowing creating a data channel toward the CDCS (server). In particular, it is an abstract component that allows encapsulating SOAP and RESTful web services, where the first is an XML-based protocol that uses service interfaces to expose the business logic, and the second is an architectural paradigm that supports different data formats including JSON, XML, HTML, and TXT. Since communication between clients may also be required, this component will be endowed with peer-to-peer networking capabilities, possibly acting as a relay between other clients and the server(s).

(ii) Native Networking API. This component is inherent to each mobile operating system platform, and it is the one providing the actual establishment of end-to-end connections between client and server.

3.2. Cloud Data Collection Server. In the context of crowdsensing applications, the main goal of the server component, which may physically consist of a single server or a server farm, is to collect all data gathered by the different clients, storing the data, and then perform all sorts of data analytics to provide the administrator or clients themselves with a summary of the most relevant information. In addition, the server allows defining and automating some of the data collection tasks. For example, the administrator can create new tasks, and these can be deployed to clients either automatically or manually; an example of this can be the collection of the noise levels for a given target area during a given period of time. Figure 5 shows our proposed architecture for the CDCS, which includes four components: Server Interface Manager (SIM), Server Task Manager (STM), Server Data Manager (SDM), and Server Communications Manager (SCM). Notice that each of these components includes a controller. Such controllers have a critical function in the scope of our architecture, as it is the communication between adjacent controllers that allows the different components to work together, similarly to the situation at the client side.

Compared to clients, CDCS elements have much greater processing and storage capabilities. Thus, data are typically processed for a better understanding through different statistical techniques (data mining). Also, the management interface is usually web-based, allowing the administrator to easily manage, visualize, and share large amounts of data.

Depending on network scalability requirements, servers may work in either centralized, distributed, or cloud-based environments. The latter allows benefitting from deployment facilities, reduced cost, and optimized resource usage, thereby minimizing infrastructure requirements.

Concerning available technologies, server solutions may rely on a wide range of platforms, from distributed architectures in the cloud, such as Amazon Web Service (AWS) infrastructure services (EC2 and S3) [41] and Google Cloud Messaging (GCM) [42], to open source approaches such as Apache Tomcat [25, 43–45], BPEL4People [46, 47], WS-HumanTask, and JBoss JBPM [15].

Below we describe in more detail the different components at the server side.

3.2.1. Server Interface Manager (SIM). The Server Interface Manager is responsible for the interaction between user and system for task and data handling. It includes two components: the Server User Interface and the Interface Controllers.

(i) Server User Interface. It allows the user to interactively manage and schedule sensing tasks. It also supports the visualization of charts relative to sensed data. Both these actions are performed using a graphical interface that is in general web-based, meaning that the system manager can operate remotely.

(ii) Interface Controller. This is the component actually in charge of communicating with other components to meet the service requirements. An example is the programming of a sensing task, where the Interface Controller coordinates with task controllers for task planning and dissemination and with the data controller for handling data storage. In addition, it also provides application programming interfaces (APIs) to allow developers to participate in the development of different crowdsensing applications and services.

3.2.2. Server Task Manager (STM). Task Management is one of the main components at the server side according to our proposed architecture, being responsible for the planning,

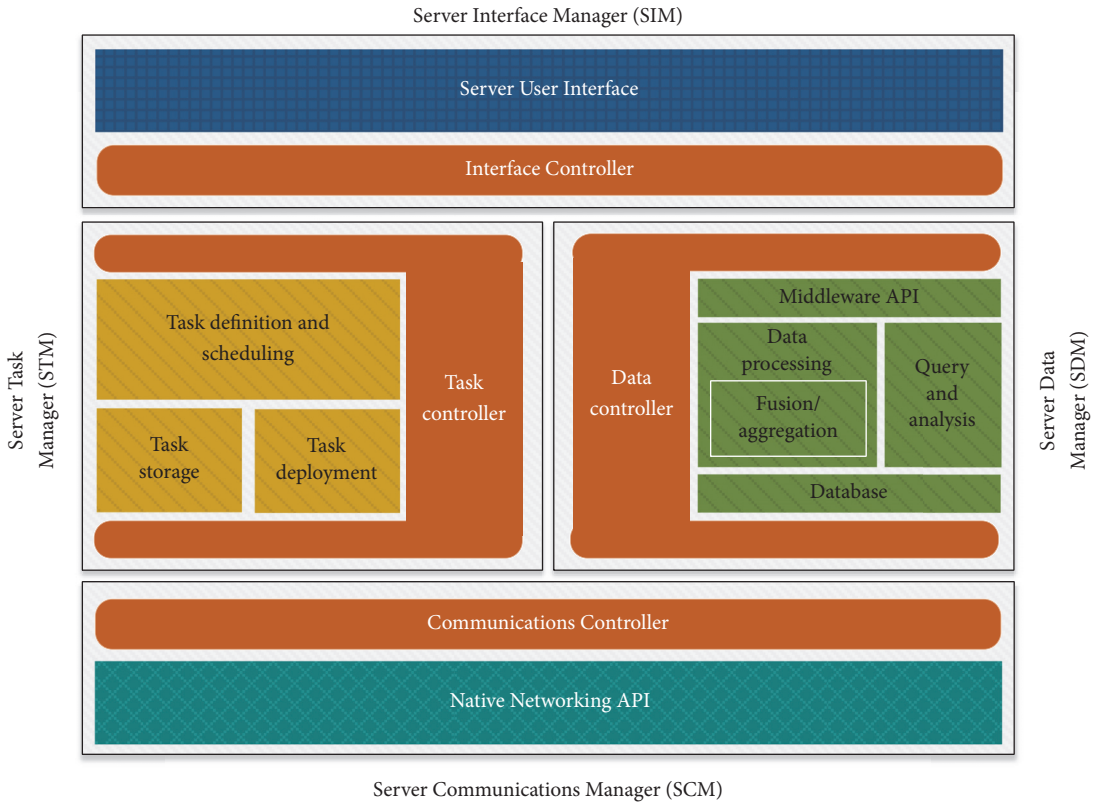

Figure 5: Cloud Data Collection Server components.

scheduling, and pushing of crowdsensing tasks. Tasks can be deployed to mobile devices either manually or automatically, and in general they rely on a system-specific language that typically differs from one solution to another due to lack of standardization. It is also worth highlighting that most implementations rely on open source tools.

The subcomponents that integrate the STM are the following.

(i) Task Controller. It works as a handler, providing the functionality required by the Server Task Manager. Typically it must attend administrator requests and may push new scheduled tasks onto clients. It can also make use of learning or approximation algorithms that optimize data collection in order to minimize energy/resource consumption at the client side. Additionally, this subcomponent includes classes and methods to start, stop, and configure the different tasks. Finally, it provides the services and functions required to interact with the other controller components. To contact clients, some implementations are based on Publish/Subscribe approaches where a server (or servers) provides a set of services to users. Additionally, in many of these Publish/Subscribe systems, the server can take intermediary functions where publishers send the messages to such intermediary server (broker), and the subscribers subscribe to information considered to be of interest, thus making this server responsible for handling the filtering, storage, and management toward the subscribers.

(ii) Task Definition and Scheduling. Among its features we can find the allocation of time and frequency of sensing, the number of mobile devices to be enabled for data collection, and the characteristics of the sensor to monitor, among others.

(iii) Task Deployment. It allows the deployment of tasks to MSCs, which can be a mere set of instructions interpreted by the existing applications. To support this option, a language defined by the application is often used, and it is typically based on SQL, XQUERY, or XML. Alternatively, a new application/component is pushed to the mobile terminal whenever new functionalities must be supported.

(iv) Task Storage. This component is responsible for the storage of current and past tasks. Since requirements are typically low, any database system suffices. In fact, it is not necessary to rely on a standard database, being also common to use a set of files, where each file describes a single task.

3.2.3. Server Data Manager (SDM). This component is responsible for the processing, storage, and analysis of the data. It is composed of a data controller, middleware APIs, a data processing element, a query and analysis element, and a database. Below we describe in more detail each of these components.

(i) Data Controller. It offers access to the services offered by the SDM, supporting a set of algorithms or applications

that allow handling data in collaboration with the task controller, Interface Controller, and other system components. In addition, it acts as a handler for communications to/from middleware APIs.

(ii) Middleware API. A middleware API is typically an extension providing more sophisticated data processing/analysis. It can incorporate data analysis tools such as data mining, analytical libraries, or other, allowing easily handling large volumes of data. Deployments at this level can be drivers or web services that enable access to databases through JDBC or other methods, such as CUPUS [48] and CAROM [27], which additionally provide data fusion and data filtering techniques.

(iii) Data Processing. The functionality of this component is similar to the data processing made at the client (CSM). The main difference is the volume of data that has to be handled at the server side. Typically, it provides functions to filter and merge multiple streams of data, providing aggregation levels that clearly surpass those levels achievable at the client side. With this purpose, it uses techniques that require a higher level of processing, such as FSI or ECSTRA [27], among others.

(iv) Query and Analysis. This component integrates both query and data analysis functionalities. It allows, through a structured query language, accessing the resources available at the server's database. Additionally, it can rely on different analysis tools to meet the requirements of other system components.

(v) Database. This component provides a database management system that allows storing the gathered data coming from the different Mobile Sensing Clients (MSCs). In the scope of the SDM, it is mandatory since it is a basic system requirement. It should be noted, though, that the database itself is not necessarily contained in a single server, and so distributed storage environments are contemplated as well. Common database management systems include MySQL and PostgreSQL, among others.

3.2.4. Server Communications Manager (SCM). The Server Communications Manager is responsible for interacting with the different clients, having characteristics similar to the Client Communications Manager. The interaction with clients is bidirectional: we have transmission toward the client when pushing new tasks, and we have transmissions from clients when receiving sensed data.

The SCM has two main components, the Communications Controller and the Native Networking API, both of which we now detail.

(i) Communications Controller. It offers the services necessary to establish communication between the MSC and the CDCS, usually as listeners for data gathering, or starting connections when task pushing is required. Additionally, it can rely on high-level communication services like SOAP and can also have adapters for any specific protocol or method of communication used by different server components. An example can be a REST-SOAP Adapter, which receives a SOAP request and adapts it to a REST-service format.

(ii) Native Networking API. This component is the one responsible for actually communicating with client devices through the establishment of end-to-end connections. Typically, reliable TCP connections are established.

4. Analysis of Existing Proposals

In this section, we provide an analysis of the different solutions available in the literature, using the architecture in Section 3 as reference for our classification. For our study, we focused on research works published in the crowdsensing field during the past five years, with a special emphasis on smartphone-based crowdsensing solutions.

For the sake of clearness and completeness, our analysis was split into four well-defined parts: (1) general analysis, (2) client-side analysis, (3) server-side analysis, and (4) data delivery approaches. The first part presents a general analysis of the various proposals, and performs a synthesis of the different contributions in the scope of our architecture. In the second part we have addressed in more detail those proposals detailing a client-side architecture, that is, describing the CUI, CDM, and MSC components, while for the third one we detail server-side architectures, describing the SUI, STM, and SDM components. In addition, we have clearly assessed to what degree the different solutions are able to provide all the functionalities envisioned in our proposed architecture. It is worth highlighting that both client and server analysis include not only proposals specific to client/server sides, but also global solutions whenever they provide details about all the elements involved in the end-to-end interaction. Finally, we have classified those solutions by providing details about the communications system defined for interactions between clients and server and also about supported topology, selected technology, and other relevant features. Again, for this data delivery analysis, any proposal providing enough details was included, no matter how broad or how specific was the proposal itself.

4.1. General Analysis. In our general analysis of the different crowdsensing solutions, we have classified information based on three parts. In the first one we provide generic information about the different proposals, in the second one we describe aspects related to security/privacy and energy consumption, and finally we provide a summary of contributions for each proposed architecture. This classification and characterization is presented in Table 2.

4.1.1. General Features. With regard to the *general features*, we found that the vast majority of solutions propose an integral solution to the sensing tasks at both client and server sides. Other solutions propose a specific middleware to help in the tasks of data collection and processing.

Concerning the strategy adopted for data collection, most proposals opted for a participatory approach for data sensing where users are fully aware of the data collection process, and

Table 2: Classification of the different technologies according to the proposed architecture.

Publication	Year	Proposal	Type	Recollection strategy	Target applications	# components levels	Privacy	Energy	MSC				Tx	CDCS			
									CIM	CDM	CSM	CCC		SCM	SDM	STM	SIM
PRISM [24]	2010	Platform	Prototype	Both	Heterogeneous	3	X	X	★	☆	☆	☆		☆	☆	☆	☆
Ear-Phone [5]	2010	Client-server	Real/simulations	Opportunistic	Environment monitoring	2		X	☆	☆	☆	☆			★	☆	☆
Anonysense [6]	2011	Framework	Prototype	Opportunistic	Environment monitoring	3	X	X	☆	☆	☆	★		★	★	★	★
Medusa [25]	2012	Framework	Prototype	Participatory	Heterogeneous	2	X		★	☆	☆	★		★	☆	★	☆
Pogo [49]	2012	Middleware	Real imp.	Participatory	Heterogeneous	2	X	X	☆	☆	☆	★		★	☆	☆	☆
DAM4GSN [50]	2012	Architecture	Real imp.	Participatory	Heterogeneous	3		X	★	☆	★	★		★	★	☆	☆
MECA [26]	2012	Middleware	Prototype	Participatory	Heterogeneous	3				☆	☆	☆		★	★	★	☆
StressSense [14]	2012	Framework	Real imp.	Participatory	Healthcare	1			★	☆	☆	☆		☆	★	☆	☆
CrowdITS [9]	2012	Framework	Real imp.	Participatory	Transportation and urban sensing	3		X	★	☆	☆	★		★	★	★	★
SmartCity [10]	2013	Framework	Real imp.	Participatory	Transportation and urban sensing	4			☆	☆	☆	☆		★	★	☆	☆
ILR [51]	2013	Scheme	Simulations/real imp.	Participatory	Location Services	3	X		☆	☆	☆	☆		★	☆	☆	☆
CAROM [27]	2013	Framework	Real imp.	Participatory	Heterogeneous	3		X	★	☆	★	☆		☆	★	☆	★
SoundOfTheCity [7]	2013	Client-server	Real imp.	Both	Environment monitoring	2		X	★	☆	☆	☆		☆	★	☆	☆
MoPS [8]	2013	Middleware	Prototype/real imp.	Both	Environment monitoring	3		X	★	★	☆	★		☆	☆	★	☆
NoiseNYC [16]	2013	Framework	Real imp.	Participatory	Healthcare	3				☆			☆	★	★	☆	☆
Vita [15]	2013	System	Real imp.	Participatory	Healthcare	2		X	★	☆	☆	★	☆	★	☆	☆	☆
Matador [52]	2013	Framework	Simulations	Both	Location services	—		X	☆	☆	☆	★	☆	★	☆	★	★
Usense [53]	2013	Middleware	Real imp.	Both	Heterogeneous	3	X	X	★	★	☆	★		★	☆		☆
REPSense [11]	2013	Applications	Real imp.	Opportunistic	Environment monitoring	2			☆	☆	☆	☆				☆	
BeC3 [46]	2013	System	Real imp.	Opportunistic	Heterogeneous	3		X	★	☆	★	★		★	☆	★	★

Table 2: Continued.

Publication	Year	Proposal	Type	Recollection strategy	Target applications	# components levels	Privacy	Energy	MSC				Tx	CDCS			
									CIM	CDM	CSM	CCC		SCM	SDM	STM	SIM
McSenseFoster [12]	2013	System	Real imp.	Participatory	Transportation and urban sensing	3			☆	☆	☆	☆	☆	☆	☆	☆	★
NoizCrowd [18]	2013	Components	Real imp.	Participatory	Healthcare	4			☆	☆	☆	☆		★	★	☆	☆
PLUS [13]	2014	Framework	Real imp.	Participatory	Transportation and urban sensing	2		X	★	☆	☆	☆		☆	☆	☆	☆
MOSDEN [28]	2014	Middleware	Real imp.	Participatory	Heterogeneous	3		X	★	★	★	★		★	☆	★	☆
SenseDroid [54]	2014	Framework	Prototype/real imp.	Both	Location services	3		X	★	★	★	☆			★	☆	
SenSocial [19]	2014	Middleware	Real imp.	Both	Social recommendation	3	X	X	★	★	☆	★		★	★	★	☆
AlienvsMobile [51]	2014	Client-server	Prototype/real imp.	Participatory	Location services	3	X		★	☆	☆	☆		☆	☆	☆	☆
SaaS [44]	2014	Framework	Description	Participatory	Heterogeneous	3				☆		☆		★	★	☆	☆
CUPUS [48]	2014	Middleware	Prototype/real imp.	Both	Heterogeneous	3		X	★	☆	★	☆	☆	☆	☆	☆	☆
BLISS [48]	2014	Framework	Simulations	Participatory	Social-budget	—			☆	☆	☆	☆				☆	☆
GPS-Less [55]	2014	System	Real imp./simulations	Both	Transportation and urban sensing	2		X	☆	☆	☆	☆		★	☆	★	☆
MCS-Space [21]	2014	System	Real imp.	Participatory	Social-general	3		X	☆	☆	☆	☆		★	★	★	☆
SPREAD [56]	2014	Algorithm	Prototype	Participatory	Location services	—			☆	☆	☆	★				☆	☆
Map++ [57]	2014	System	Real imp.	Participatory	Transportation and urban sensing	—		X	☆	☆	☆	☆			☆	☆	☆
JoiPolices [58]	2014	System	Simulations	Participatory	Heterogeneous	3	X		☆	☆	☆	☆		★	☆	☆	☆
CrowdRecruiter [59]	2014	Framework	Real imp.	Both	Location services	—		X		☆		☆		☆	☆	★	☆
Neighbor [60]	2014	Middleware	Simulations	Opportunistic	Heterogeneous	—		X		☆	☆	☆	★	☆		☆	☆
LineKing [22]	2014	System	Real imp.	Both	Social recommendation	2		X	★	☆	☆	☆		★	☆	☆	☆
GROPING [61]	2014	Application/Fr	Prototype	Participatory	Location services	2		X	★	☆	☆	☆		☆	★	★	☆
EasyHarvest [45]	2014	Framework	Prototype	Opportunistic	Heterogeneous	2			★	☆	☆	★		★		☆	☆

Table 2: Continued.

Publication	Year	Proposal	Type	Recollection strategy	Target applications	# components levels	Privacy	Energy	MSC				Tx	CDCS			
									CIM	CDM	CSM	CCC		SCM	SDM	STM	SIM
WiFIScout [62]	2014	System	Real imp.	Participatory	Social recommendation	2			☆	☆	☆	☆		☆	☆		
QoS-Constrained [63]	2014	Framework	Simulations	Both	Heterogeneous	2					☆	☆		☆	☆	★	
Ecosystem [47]	2015	System	Real imp.	Participatory	Social-general	4		X	★	☆	☆	★		★	☆	★	☆
MCSaaS [43]	2015	Framework	Real imp.	Both	Heterogeneous	4		X	☆	☆	★	★		★	★	★	★
COUPON [64]	2015	Framework	Simulations	Opportunistic	Heterogeneous	—		X	☆	☆	☆	☆	★	☆			
ADTS [65]	2015	Application	Simulations	Participatory	Location services	1			☆	☆	☆	☆					
Anonymity [66]	2015	Framework	Prototype/real imp.	Participatory	Transportation and urban sensing	2	X	X	☆	☆	☆	☆	★		☆	☆	☆
TYT [17]	2015	Framework	Real imp.	Participatory	Healthcare	3	X		★	☆	☆	☆		☆	☆	☆	★
QOATA [67]	2015	Application	Simulations	Participatory	Heterogeneous	—				☆	☆	☆					
NeCoRPIA [68]	2015	Protocols	Simulations	Opportunistic	Heterogeneous	—				☆		☆	★	☆		☆	
QoSMCS [69]	2015	Framework	Simulations	Opportunistic	Heterogeneous	3		X		☆	☆	☆	★	☆	★	☆	☆
FlierMeet [70]	2015	Framework	Real imp.	Participatory	Transportation and urban sensing	3			★	☆	☆	☆		☆	☆	☆	☆
PRESM [23]	2015	Scheme	Real imp.	Participatory	Transportation and urban sensing	3	X		★	☆	☆	☆		☆	☆	★	☆
SmartRoad [71]	2015	Framework	Real imp.	Participatory	Transportation and urban sensing	2		X	★	☆	★	★		★	★	★	☆
MCSgame [72]	2015	System	Simulations	Participatory	Location services	2		X		☆	☆	☆		★	☆	☆	☆
RemoteCloud [73]	2015	—	Simulations	Participatory	Transportation and urban sensing	—					☆	☆	★		☆	☆	☆
MDPPs [74]	2015	Middleware	Real imp.	Opportunistic	Location services	3			★	☆	☆	☆		★	★	★	☆
RuPS [75]	2015	Framework	Real imp.	Participatory	Location services	4			☆	☆	☆	☆		☆	★	☆	★
EffSense [76]	2015	Framework	Simulations	Opportunistic	Heterogeneous	—		X		☆	☆	☆	★		☆	☆	
PLP [77]	2015	Scheme	Simulations	Opportunistic	Heterogeneous	—	X			☆	☆			★	★	☆	
Sahyog [78]	2015	Middleware	Real imp.	Participatory	Heterogeneous	3	X		★	☆	☆	★		☆		☆	
Context [79]	2015	Middleware	Simulations	Opportunistic	Social recommendation	—		X		☆	☆	☆	★		☆	☆	
MoreWithLess [80]	2015	Framework	Real imp.	Participatory	Heterogeneous	4		X		☆					★	★	
Sparse [81]	2016	Framework	Real imp.	Participatory	Heterogeneous	3	X			☆					★	★	

they actively participate in that process. Other approaches, however, prefer using opportunistic systems that operate in a more autonomous manner, gathering information in the background at appropriate times; finally, a few proposals combine these two approaches to achieve a more complete functionality.

Regarding the target applications addressed in the different works, we found that the majority of the proposals are flexible enough to embrace heterogeneous applications; that is, they can adapt to generic sensing tasks, although we can also find proposals that are specific to transportation and urban sensing environments, and to a lesser extent to health, social, and other environments.

Finally, with respect to the number of differentiated elements defined for each proposed architecture, we found that there are significant differences among authors. For instance, [18, 43, 47] split their proposed functionality into four different levels, similarly to our proposal. In particular, NoizCrowd [18] defines an architecture based on four components which are data gathering, data storage, noise modeling, and data analytics/visualization. SmartCity [47] also defines a four-element architecture composed of social networks, Ubiquitous Sensors, a Mobile Context-Aware Platform, and the Cloud Platform. MCSaaS [43] defines four core submodules, namely, Cloud Broker, Orchestrator, Customization Service, and Deployment Manager. In general, most proposed architectures only defined two or three levels, as is the case of [10], which defines a generic Publish-Subscribe communication with three roles (producers, services providers, and consumers), along with Analytics Components.

4.1.2. Privacy and Energy Issues. In general, the success of mobile crowdsensing applications is dependent on how each solution addresses user concerns about his/her own privacy. Energy consumption is another critical issue, as applications draining a significant amount of battery power will be rejected by most users. So, both energy and privacy issues are relevant in the scope of crowdsensing solutions, the reason why they have been addressed by different researchers.

Our analysis has shown that most studied proposals have addressed energy efficiency issues, while only some of these have introduced mechanisms to mitigate security and privacy concerns. In fact, we find that very few solutions [6, 19, 24, 53, 66] actually account for both privacy and energy efficiency issues. We now proceed to discuss these prominent solutions in more detail.

PRISM [24] supports privacy though a registration process on a PRISM server for each enabled terminal. The registration is maintained by software and it expires within a given period of time. When the registration period expires, terminals wait for a random time and proceed to register again. With regard to energy consumption, PRISM maintains a control of energy consumption on mobile phones through its prism sandbox, which is able to perform coarse-grain power monitoring.

Anonysense [6] uses a server that is responsible for registering and authorizing mobile phones. During registration, Anonysense installs its software along with the IP addresses and certificates for its task service and report service. Concerning energy consumption, the tasks can be divided into two suboperations: sensing and signing. In the first, the RogueFinder application is used to detect rogue APs in a given area, while the ObjectFinder application attempts to find a specific Bluetooth MAC address. The second group addresses whether a data report contains sensitive data. Additionally, it estimates the energy cost associated with these operations.

Usense [53] includes a component for securing communications. Additionally, it manages user preferences in terms of resource and privacy restrictions. These features are processed through the sensing agent, which is an application deployed on the device itself. In addition, Usense's middleware is able to save energy using a mechanism that avoids taking measurements in those areas where it already has enough data, or when the phenomenon is mostly invariant.

SenSocial [19] has a module for privacy management control which allows managing policies regarding the type and level of granularity of sensed data, deciding what will be stored and made available to the different middleware components. SenSocial uses filtering rules for maintaining energy efficiency, thereby restricting transmissions only to those cases passing the set of defined rules. Also, SenSocial discriminates the energy consumption associated with the accelerometer sensors, microphone, GPS, Bluetooth, and WiFi.

The last proposal in this group is Anonymity [66], which proposes an anonymous data reporting protocol for participatory applications. The idea is that the protocol avoids including identification information that can be vulnerable. The anonymous data protocol is divided into two stages: the first is a slot reservation stage (scheme based on public key encryption), while the second one is a data submission stage (scheme based on an XOR operation). Through comparison against a similar study, authors show how it is able to improve data submission performance. With regard to energy consumption, the smartphone's battery values are measured using a multimeter. It also presents an analysis of the energy overhead associated with data submission.

4.1.3. Analysis of Contributions for Each Proposal. The main goal of this survey is to assess the actual contributions made by the different authors taking as reference the architecture proposed in Section 3. So, the last part of Table 2 (columns MSC, Tx, and CDCS) provides a first insight into the actual contribution made by the different components at the client (MSC) and server (CDCS) sides, in addition to the end-to-end transmission process itself (Tx).

We provide a three-level classification of proposals, where a dark star means that the particular solution fulfills the expected functionality for that component, while a white star means that the solution only provides a partial fulfillment of the selected characteristics. The nonfulfillment of the characteristics of a component is represented by the absence of any star.

Overall, we can observe that the majority of the proposals are quite representative in the scope of our architecture, providing most of the expected functionalities. Nevertheless, we can also find solutions such as MOSDEM [28] and

SenseDroid [54] that focus mostly on MSC-related functionality. Similarly, we can find solutions such as MCSaaS [43] that focus on the CDCS instead.

4.2. Client-Side Analysis. In this section we focus on the specific contributions to the MSC, which is the client side of our proposed architecture. To achieve it, in Table 3, we describe the features of the different proposals regarding the Client Interface Manager (CIM), the Client Data Manager (CDM), and the Client Sensor Manager (CSM). Notice that we excluded the Client Communications Manager (CCM) from this section, as it will be addressed separately in Section 4.4. Also notice that the table is split into two sections, being that proposals in the upper section are client-specific, meaning that the publication only describes the client side of the crowdsensing architecture, while proposals in the bottom section describe both client and server sides.

Concerning the CIM, we found that most of the solutions provide a graphical User Interface designed for the Android operating system, thus typically adopting the Java language for development. In fact, only a few solutions such as LineKing [22], TYT [17], and DAM4GSN [50] focused on other operating systems. Also, most of the proposals allow the user to have access to an administrative interface in order to have control over sensing tasks.

With regard to the CDM we observe that, in general, most available solutions resort to plugins or external libraries in order to simplify their processing, query, and storage tasks on the device by reusing existing software. In particular, different techniques and algorithms are adopted mostly to support the data collection procedure including spatiotemporal area calculation and programming algorithms. The spatiotemporal coverage of an area refers to the amount of time and space needed to properly sense that area according to the target task, Usense [53] being the most widely used. Also, we found that although several solutions provide data analytics within the mobile device itself, such functionality is seldom combined with the use of plugins.

Among solutions integrating plugins, we would like to highlight solutions, such as DAM4GSN [50], MOSDEN [28], and CAROM [27], that use open source GSN technologies for IoT. In particular, CAROM [27] uses a plugin where, among other functionalities, it incorporates Open Mobile Miner (WMO), which is an open source solution that allows performing data analysis on the mobile terminal. Similarly, SenSocial [19] uses a plugin providing an agent able to retrieve data from both Facebook and Twitter, and its process is based on joining online social networks (OSNs) that provide a physical context data stream. In addition, we found that few solutions include a broker functionality. We also found that there is a balance between the approaches preferring pushing contents onto the servers and solutions that prefer the server to pull contents instead.

With regard to data processing, we find that few solutions perform aggregation-fusion on the mobile device, as opposed to data filtering, whose support is quite common. Finally, with regard to the Client Sensor Manager, in general, the different proposals available make use of generic sensors that are internal to the mobile devices, offering in a few cases support for external sensors. There is also evidence of applications using external sensors or multimedia stream processing before sending the streams to the server (see, e.g., StressSense [14] and REPSense [11]).

Finally, regarding the adoption of virtual sensors, only a minority of the proposals studied do so. In particular, options such as DAM4GSN [50], MOSDEN [28], and CAROM [27] relied on an adapted version of GSN [40], while other proposals like SenseDroid [54], CUPUS [48], and SmartRoad [71] provide their own virtualization solutions.

4.3. Server-Side Analysis. In this section we will focus instead on the server side, which in the scope of our proposed architecture takes the name "Cloud Data Collection Server" (CDCS).

Table 4 describes the features of server-related proposals. Similarly to the previous section, the table is split into two parts, being that proposals in the upper part are server-specific (the publication only describes the server side of the crowdsensing architecture), while proposals at the bottom section are complete ones, describing both client and server sides; obviously, since client-related details were already presented above, in this section we only focus on server-related issues.

In general, we observe that most of the proposals provide a web interface for management and result presentation purposes, and most of them also provide data management and data sharing functionalities. The technologies used in these proposals are generally open source solutions like Apache, Java, and PHP, among others, and many of them use a database manager such as MySQL and PostgreSQL. Also, there is evidence that many proposals rely on a cloud infrastructure provided by Amazon [41] or Google [42].

With respect to the Server Task Manager, we find that most proposals present mechanisms to manage and deploy sensing tasks. In particular, in terms of task deployment, we find that the number of proposals adopting a push-based approach is similar to those adopting a pull-based approach.

Regarding the language used for task definition, some solutions describe tasks using specific algorithms, while others prefer using a programming language, as is the case of Pogo [49], Anonysense [6], and Medusa [25].

With respect to data management at the server, most solutions perform data aggregation similarly to client-side solutions. Some of them use intelligent data analysis techniques such as Big Data [8, 10, 43], MCDM [75], and PFISR [21], and various other statistical tools. Recent research works [80–82] take advantage of the space and the time correlation between the discovered data of different subareas with the aim of reducing the number of tasks required for the target purposes. Wang et al. [81, 82] present a solution called sparse MCS framework that uses inference algorithms to ensure the quality of the data after being collected. Instead, Xu et al. [80] describe a framework that uses four states (data structure conversion, base training, sampling, and reconstruction). It relies on programming algorithms to create a baseline dataset using the K-SVD algorithm, while for the reconstruction the Orthogonal Matching Pursuit recovery algorithm is adopted. In both cases, the intention is to produce a global saving

Table 3: Classification of the different technologies according to the proposed client architecture.

Publication	Client interface manager (CIM)				Client Data Manager (CDM)								Client Sensor Manager (CSM)		
	Interface	Admin	OS	Technology	Characteristic	Plugins/external libraries			Data processing		Query	Local store	Type of sensor	Virtual sensor	Preprocessing
						Real time	Sensing task	Data analytics	Broker/plugin	Filter	Aggregate				
PRISM [24]	X		Windows mobile	C# and C++	PRISM's sandbox	Sent	Push-pull					X	X	Generic/multimedia		
DAM4GSN [50]	X	X	Android/IOS	Java	XML-based/GSN	Store and send	Push-pull		Both	X		X	X	Generic and external	X	X
StressSense [14]	X		Android	Java, C and C++	GMMs and simulate	Send	Pull		Plugin				X	External microphone		X
Usense [53]	X	X	Android	Java-BlueZen	XML-based/spatiotemporal	Store and send	Push		Plugin	X	X	X	X	Generic/multimedia		X
REPSense [11]	X		Android		Scheme divide/merge	Store and send				X		X	X	GPS, ambient light, air pressure, accelerometer		X
PLUS [13]		X	Android		Markov Predictor Model	Store and send		X		X			X	GPS		X
MOSDEN [28]	X	X	Android	Java	XML-based/GSN	Store and send	Push-pull	X	Both	X		X	X	Generic and external	X	X
SenseDroid [54]			Android		Spatiotemporal	Store and send			Broker	X	X		X	Generic and external	X	X
AlienvsMobile [39]	X		Android	Java	Area coverage	Send	Pull							GPS, barometric pressure		X
CUPUS [48]	X		Android		CUPUS MIOs	Store and send	Push-pull			X		X	X	Generic and external	X	X
BLISS [20]				Simulated	Simulated/online learning	Send		X		X		X		N/A simulated		
SPREAD [56]				Simulated	Simulated/area of interest	Send				X		X		GPS and simulated		
MAP+ [57]	X		Android		DBSCAN	Store and send	Pull			X			X	External GPS		X
ADTS [65]				Simulated	ADTS algorithm	Send		X		X				Generic		
FlierMeet [70]	X		Android		STA grouping	Send	Pull							GPS, light sensor, accelerometer, magnetometer		X
MDPPs [74]		X	Android		Spatiotemporal coverage (MDPPs)	Stored and sent	Push	X				X	X	Generic sensor		
EasyHarvest [45]	X	X	Android	Java for Android	Spatiotemporal coverage	Store and send	Binary push	X		X		X	X	Generic sensor		

Table 3: Continued.

Publication	Client interface manager (CIM)				Client Data Manager (CDM)								Client Sensor Manager (CSM)		
	Interface	Admin	OS	Technology	Characteristic	Plugins/external libraries				Data processing		Query	Local store	Type of sensor	Virtual sensor	Preprocessing
						Real time	Sensing task	Data analytics	Broker/plugin	Filter	Aggregate					
Sahyog [78]	X		Android	Java	Query format	Store and send	Push-pull					X	X	GPS, accelerometer		
WiFIScout [62]	X	X	Android		Read WiFi	Store and send	Pull			X			X	GPS/WiFi		
Ear-Phone [5]	X		Symbian	Java	Spatiotemporal coverage	Send	Pull			X		X	X	GPS microphone		X
Medusa [25]		X	Android	Java SMS and MMS	MedScript XML	Send	Push	X	Broker	X			X	GPS, multimedia		X
CrowITS [9]	X	X	Android/IOS		Plugin-based	Store and send	Push-pull		Plugin			X		GPS		
CAROM [27]	X	X	Android	Java/GSNLite	XML-based/GSN	Send	Push-pull	X	Plugin (OMM)		X	X	X	Generic and external sensor	X	X
SoundOfTheCity [7]	X	X	Android	Java		Store and send	Pull					X	X	GPS, microphone		X
MoPS [8]	X	X	Android	Java	Broker Mios		Push	X	Broker	X		X		External pollution		X
Vita [15]	X	X	Android	Java	XML-based	Send	Push		Broker	X		X	X	Generic and multimedia	X	
Matador [52]	X	X	Android		XML-based spatiotemporal	Send	Pull	X		X			X	GPS		X
SenSocial [19]	X		Android	Java	XML-based/plugin OSN	Send	Push-pull	X	Both	X	X	X	X	GPS, accelerometer, microphone, social		
MCSinSpace [21]	X	X	Android			Send	Push			X		X	X	GPS signal analysis		X
LineKing [22]	X		Android/IOS		Wait-time algorithm	Send	Pull	X		X		X	X	GPS, accelerometer		
Ecosystem [47]	X	X	Android	Android app	XML-based	Send	Push	X		X			X	Generic and external sensor		X
TYT [17]	X	X	Android/IOS	Android and IOS apps		Send	Pull					X	X	Heart rate, blood pressure, oxygen saturation		
PRESM [23]			Android	RSS map	Map generation	Store and send	Push	X		X				GPS locations		X
SmartRoad [71]	X	X	Android	Java	Detection and identification learning algorithm	Store and send	Push		Plugin			X		GPS, power sensor	X	X

Table 4: Classification of the different technologies according to the proposed server architecture.

Publication	Server Interface Manager (SIM)				Server Task Manager (STM)				Server Data Manager (SDM)				
	Interface	Manager	Shared	Technology	Language	Task	Cloud	Automatic task	Sensing task	Middleware	Data processing: Filter	Data processing: Aggregate	Query and analysis	Database
Anonysense [6]	Web	X	X and maps	Ruby and small HTTP Severs	AnonyTL/Ruby	Authentication servers		X	Push/pull	Privacy distribution	X		Query	SQLite3
Pogo [49]	Web	X		JavaScript-Openfire XMPP Pub/Sub	Scripts-Rhino	Generic		X	Push	Multibroker	X	X	Query	SQL
MECA [26]	Web		X		Edge task	Generic	X		Pull	Analytics library and multibroker		X	Query and analysis	SN and device
SmartCity [10]	Web		X	XMPP Pub/Sub		Learning algorithm	X	X	Push	Big Data, data mining Casandra/s4	X	X	Query and analysis	NoSQL
ILR [51]	Web	X		Java/J2EE and Glassfish Application Server	ILR scheme	Location reliability algorithm			Push	Simulations NS2 Real Traces	X		Query and analysis	Derby
NoiseNYC [16]						Generic				3D tensor Kriging, heatmaps, and others		X	Query and analysis	S/N and device
BeC3 [46]	Web	X	X	C, Java/XMPP Pub/Sub, D-LITe Cloud	D-Lite/Python BPEL WS-CDL	Generic	X	X	Push/pull			X	Query	S/N and device
McSense [12]	Web	X	X	Java Servlet Ajax Web-Apache Spring		Generic	X	X	Push		X		Query and analysis	PostgreSQL
SaaS [44]	Web	X	X			Generic	X	X	Push		X	X	Query and analysis	S/N and device
GPS-less [55]	Web			JavaScript		Coverage model algorithm		X	Push			X	Query and analysis	S/N and PostgreSQL
JoinPolices [58]						Budget efficiency algorithm			Push	MatLab		X	Query and analysis	
MCSaaS [43]	Web	X	X	Java, Python/Apache Tomcat, Google Cloud Messaging (GCM)	XML task	Generic	X	X	Pull	Cloud broker Big Data	X	X	Query and analysis	S/N and device
GROPING [61]				Amazon Mechanical Turk (AMT)		Location estimation algorithm	X		Pull		X	X	Query and analysis	S/N

Table 4: Continued.

Publication	Server Interface Manager (SIM)				Server Task Manager (STM)				Server Data Manager (SDM)				
	Interface	Manager	Shared	Technology	Language	Task	Cloud	Automatic task	Sensing task	Middleware	Data processing: Filter	Data processing: Aggregate	Query and analysis	Database
Qoata [67]					QOATA scheme simulated	Online learning and task allocation algorithm						X		
MCS game [72]						Q-learning algorithm	X		Push	Nash equilibrium (NE)			Query	
RuPS [75]	Web	X	X			Generic			Pull	Multi-Criteria Decision Making (MCDM)		X	Query and analysis	S/N and device
NoizCrowd [18]			X			Spatial and temporal algorithm	X		Pull	Big Data		X	Query and analysis	SciDB and SPARQL
QoS-Constrained [63]						Spatial temporal coverage			Pull	QoS: Min, Max, utility	X	X		
CrowdRecruiter [59]						Spatial temporal coverage		X	Pull		X	X		S/N
PLP [77]						Learning algorithm			Pull	RSS fingerprint	X	X	Query and analysis	S/N
MoreWithLess [80]						Spatial temporal correlations		X	Pull		X	X	Query and analysis	
Sparse [81]						Spatial temporal correlations			Pull		X	X	Query and analysis	
Ear-Phone [5]	Mobile			PHP script and Java		Generic			Pull	GPS MGRS converter		X	Query	MySQL
Medusa [25]	Web			Apache, PHP, Java, and AMT	MedScript/Python	Generic	X		Push	Stage Library	X		Query	MySQL and device
CrowITS [9]	Web	X	X	Google C2DM and PHP-Google Maps API		Generic	X	X	Push/pull		X	X	Query and analysis	MySQL and device
CAROM [27]	Web	X	X	AMT EC2 and GSN	Task-XML/GSN	Generic	X	X	Push/pull	Data mining/FSI fussy	X	X	Query and analysis	S/N and device
SoundOfTheCity [7]	Web		X	JavaScript		Generic			Pull	Media streaming and encoding		X	Query and analysis	MySQL
MoPS [8]	Web			Java and Google Cloud Messaging (GCM) Pub/Sub		Generic	X	X	Push/pull	Data mining		X	Query	S/N
Vita [15]	Web	X	X	Apache Tomcat and JBoss and (AWS) EC2 AND S3	Task BPEL/ BPEL4People: ODE and jBPM	Genetic algorithm and K-means	X	X	Push			X	Query	S/N
Matador [52]	Web	X	X		Matador task	Spatial temporal algorithm		X	Pull	Adaptive sampling algorithm		X	Query	S/N
SenSocial [19]	Web		X	PHP script/Pub-Sub (MQTT)	XML	Task OSN		X	Push/pull	OSN and filter aggregated	X		Query and analysis	MongoDB and device
MCSinSpace [21]	Web	X		Mapping and tomographic reconstruction algorithms		Generic	X	X	Push	PFISR, Kalman filter, and interpolates	X	X	Query and analysis	S/N

Table 4: Continued.

Publication	Server Interface Manager (SIM)					Server Task Manager (STM)					Server Data Manager (SDM)			
	Interface	Manager	Shared	Technology	Language	Task	Cloud	Automatic task	Sensing task	Middleware	Data processing		Query and analysis	Database
											Filter	Aggregate		
LineKing [22]	Web			Apache HTTP and AWS EC2		Generic	X	X	Pull	Wait-time estimator	X		Query	MySQL and device
Ecosystem [47]	Web	X	X	Amazon EC2 M1, Ubuntu, and Apache Tomcat	Task BPEL XML	Generic	X	X	Push/pull		X	X	Query and analysis	S/N
TYT [17]	Web	X	X	PHP Laravel		Generic	X		Pull	Algorithm TYT		X	Query	MySQL
PRESM [23]	Web		X			Algorithms CS-based and RSS		X	Push	RSS Map generation	X		Query	S/N
SmartRoad [71]	Web			Java, Django, and Python plugin/Google Maps		Learning algorithm		X	Push	Heatmap, 3D view		X	Query and analysis	MySQL and device

on detection costs (power consumption, network resources) while ensuring the overall data quality.

As output, data can be presented in different formats, the use of heatmaps being a representative example when sensing that information is geolocated.

4.4. Data Communications Issues. We conclude our analysis of the current crowdsensing literature by focusing on client-server communication solutions. Notice that, since communications simultaneously involve clients and servers, we address communication issues jointly in this section.

Table 5 summarizes the main communication characteristics associated with the different proposals. We have also surveyed the metrics used by each proposal for performance analysis and classified them according to their scope as generic, QoS, and scalability. As *generic* performance metrics we refer to those proposals addressing network performance in terms of packet delivery ratio, end-to-end delay, and transmission overhead, among others. *QoS* issues are associated with data acquisition, and they attempt to avoid the fact that the delivery of massive data (data without processing) directly from the source negatively impacts network traffic and the energy consumption of mobile devices. Concerning *scalability*, authors assess the capability of the infrastructure in terms of adaptability to an increasing number of sensing tasks and terminals to determine if it is able to adapt to both small and large deployments. Under the scalability concept we have also considered the elasticity of these services (middleware) to manage changes.

Notice that Table 5 is clustered into four different parts according to the scope of the proposal: T refers to those proposals only addressing transmission issues, C refers to proposals centered on the client side, S refers to proposals centered on the server side, and G refers to global solutions.

Concerning communication technologies used, a large number of proposals relied on WiFi and Cellular communications, although we can also find proposals that rely instead on Bluetooth due to its flexibility and low consumption features. Additionally, we find that most solutions opted for either a centralized topology or a distributed topology, with only a reduced number of proposals choosing a hybrid approach. Regarding the networking approach, most solutions adopt RESTful services based on HTTP or make use of the XML format.

Focusing now on the performance metrics addressed by each proposal, most solutions made a generic performance analysis (delivery delay, data rate, etc.). However, very few solutions addressed QoS and scalability issues. For instance, we can find solutions such as GCM [9] that address scalable services in the cloud, others that address scalability in the context of the Publish/Subscriber paradigm [8], and yet others that relate it to broker collaboration [54], but none of these actually assess performance in the scalability context.

Regarding proposals evaluating Quality of Service performance, they typically perform such evaluation in terms of task allocation and coverage optimization in the target area. For instance, proposals such as JoinPolices [58] evaluate the impact and the performance of task execution based on incentive policies, while QoSMCS [69] defines an ad hoc method for the evaluation of QoS in the context of mobile crowdsensing services based on Petri networks.

Finally, regarding scalability, solutions such as PRISM [24] assess the performance achieved through comparison against other solutions. Neighbor [60] measures message diffusion performance between the mobile nodes and the data collection server. Lastly, Medusa [25] proposes a prototype able to measure in runtime the time taken to perform several individual steps associated with task executions, both on the cloud and the smartphone.

5. Open Research Issues

Based on the analysis presented in the previous sections, it becomes clear that, despite the many advancements introduced in the mobile crowdsensing field in recent years, there are still several issues that should be properly addressed for solutions to become more effective and therefore gain more widespread acceptance.

At the user's side, it becomes clear that the sensing tasks should not become a burden. Thus, any external sensors, if required at all, should be small and lightweight, have a low power consumption, and have an elegant and stylish look. Ideally, additional sensors should be progressively integrated into new smartphones either directly from the manufacturer or as pluggable modules. Power and network resource consumption are also an issue, and so smart algorithms able to correctly determine the best sampling times while avoiding intensive CPU usage are required; in terms of network resources, peer-to-peer data delivery combined with smart network selection can help at avoiding to deplete radio resources and having a negative impact in terms of traffic quotas.

From a more global perspective, further studies are required in order to assess the scalability and the QoS support of the different proposals. In particular, their impact on the end-to-end communications infrastructures should be thoroughly studied. Additionally, new algorithms should be developed to improve the processes of data collection and analysis.

6. Conclusions

Crowdsensing solutions that benefit from smartphones are proliferating due to the multiple advantages offered. Thus, it becomes important to provide a unified view of the different author contributions to detect the major areas of improvement. In this paper we address this challenge through a survey that provides the reader with an extensive review of existing smartphone-based solutions in the field of mobile crowdsensing. We start by presenting a novel reference architecture where we identify the major components at client side, server side, and the communications level. Based on our proposed architecture, we then proceed to classify the different proposals, focusing separately on the client, the server, and the communications part of each solution.

Our extensive literature analysis has shown that most proposals provide some degree of adaptability to different work environments. In addition, we found that technologies

Table 5: Classification of the different data transmission solutions according to the proposed architecture.

Publication	Scope	Communication technologies	Topology	Networking approach	Protocols/format	Performance metrics		
						Generic	QoS	Scalability
Neighbor [60]	T[1]	WiFi/Bluetooth/Cellular	Hybrid	Algorithm/one simulated		X		X
Coupon [64]		WiFi/Bluetooth	Hybrid	ERF and BSWF	TCP	X		
Anonymity [66]		WiFi	Centralized	Slot reservations stage and data submission stage	TCP	X		
NeCorpia [68]		WiFi	Distributed-hybrid	Network encoding format/Gaussian eliminations	TCP	X		
QoSMCS [69]			Hybrid	Markovian stochastic Petri NeT	HTTP-TCP		X	
RemoteCloud [73]		WiFi/Cellular	Distributed	MANET or device-to-device (D2D)	TCP	X		
EffSense [76]		Bluetooth/WiFi/Cellular	Hybrid	Publish-Subscriber	HTTP-JSON	X		
Context [79]		Bluetooth	Hybrid	Simulated	TCP	X		
PRISM [24]	C[2]	WiFi/Bluetooth/Cellular	Centralized	PRISM client and Sandbox				X
DAM4GSN [50]		WiFi/Cellular	Hybrid	GSN (SOAP and RESTful web services)	HTTP-XML			
StressSense [14]		WiFi/Cellular	Distributed	MatLab				
Usense [53]		WiFi/Cellular	Distributed	SOAP web services (RPC)	HTTP-XML	X		
REPSense [11]		WiFi/Cellular	Distributed					
PLUS [13]		WiFi	Centralized	Web services				
MOSDEN [28]		WiFi/Cellular	Distributed	GSN (SOAP and RESTful web services)	HTTP-XML	X		
SenseDroid [54]		WiFi/Bluetooth/Cellular	Hybrid	Provides libraries and APIs				
AlienvsMobile [39]		WiFi	Centralized	NS2 simulated	HTTP	X		
CUPUS [48]		WiFi	Distributed	Publish-Subscriber (MQTT-MOSQUITO)	HTTP-XML	X		
BLISS [20]				Simulated				
SPREAD [56]		S/E		Simulated				
MAP+ [57]		WiFi/Cellular	Centralized					
ADTS [65]		WiFi/Cellular	Hybrid	Simulated				
FlierMeet [70]		WiFi	Distributed	—				
MDPPs [74]		WiFi/Bluetooth/Cellular	Distributed	SOAP web services	HTTP-XML			
EasyHarvest [45]		WiFi/Cellular	Centralized	RESTful web services	HTTP-XML			
Sahyog [78]		WiFi/Cellular	Centralized	Publish-Subscriber	HTTP-JSON	X		
WiFIScout [62]		WiFi	Centralized		HTTP			
Anonysense [6]	S[3]	WiFi/Bluetooth	Hybrid	SOAP web services (SMTP/SSL)	HTTP(S)-XML			
Pogo [49]		WiFi/Cellular	Centralized	Publish-Subscriber (XMPP Openfire)	HTTP-JSON	X		

Table 5: Continued.

Publication	Scope	Communication technologies	Topology	Networking approach	Protocols/format	Performance metrics		
						Generic	QoS	Scalability
MECA [26]		WiFi/Cellular	Distributed					
SmartCity [10]			Distributed	Publish-Subscriber (XMPP)	HTTP-XML			
ILR [51]		WiFi/Bluetooth	Centralized	ILR Algorithm/simulated NS2				
NoiseNYC [16]			Distributed					
BeC3 [46]		WiFi	Distributed	RESTful web services (COAP)	HTTP-XML			
McSense [12]		WiFi/Bluetooth	Distributed	RESTful web services	HTTP-XML	X		
SaaS [44]		WiFi/Bluetooth/Cellular	Distributed	Web services	HTTP			
GPS-Less [55]		WiFi	Distributed	Approximation algorithms	HTTP	X		
JoinPolices [58]		Budget	Distributed	Simulated MatLab		X	X	
MCSaaS [43]		WiFi/Cellular	Distributed	RESTful web services	HTTP-XML	X		
GROPING [61]		WiFi	Centralized					
Qoata [67]			Centralized	Simulated				
MCS game [72]		WiFi/Cellular	Centralized					
RuPS [75]		WiFi/Cellular	Centralized	RESTful web services	HTTP			
NoizCrowd [18]			Distributed					
CrowdRecruiter [59]		WiFi/Cellular	Centralized	Simulated				
QoS-Constrained [63]			Hybrid	Simulated				
PLP [77]			Hybrid					
MoreWithLess [80]			Distributed					
Sparse [81]			Distributed					
Ear-Phone [5]		WiFi/Cellular	Centralized					
Medusa [25]		WiFi/Cellular	Distributed	MedScript-SMS-MMS	HTTPS-XML	X		X
CrowITS [9]		WiFi/Cellular	Centralized	RESTful web services (C2DM)	HTTP-JSON			
CAROM [27]		WiFi/Bluetooth/Cellular	Hybrid	GSN (SOAP and RESTful web services)	HTTP	X		
SoundOfTheCity [7]		WiFi/Cellular	Centralized	SOAP web services (RTMP)	HTTP-XML			
MoPS [8]		WiFi/Cellular	Distributed	RESTful web services (GCM)	HTTP-XML			
Vita [15]		WiFi/Cellular	Hybrid	RESTful web services	HTTP-XML	X		X
Matador [52]	G[4]	WiFi/Cellular	Centralized	RESTful web services	HTTP-XML			
SenSocial [19]		WiFi/Bluetooth/Cellular	Centralized	Publish-Subscriber (MQTT-MOSQUITO)	HTTP-JSON	X		X
MCSinSpace [21]		WiFi/Cellular	Distributed	RESTful web services	HTTP			
LineKing [22]		WiFi	Centralized	SOAP web services	HTTP			
Ecosystem [47]		WiFi/Cellular	Distributed	RESTful web services	HTTP-XML	X		
TYT [17]		WiFi/Bluetooth/Cellular	Centralized	RESTful web services	HTTP			
PRESM [23]		WiFi/Cellular	Centralized					
SmartRoad [71]		WiFi/Cellular	Centralized	SOAP web services	HTTP-XML			

[1]T: proposals addressing transmission issues.

[2]C: proposals centered on the client side.

[3]S: proposals centered on the server side.

[4]G: global solutions.

and algorithms applicable at both client and server sides have evolved significantly and are often available in a modular format, allowing other researchers to include them in their proposed solutions. Concerning improvements in the data capture process itself, we found that the main issues are the software adaptability to different types of sensors and reducing power consumption. At the server side, the most critical improvements include task generation language and procedures, the analysis and storage of data, and providing an adequate interface for task management by administrators. The communication between client and server usually makes use of technologies like SOAP and RESTful, and most solutions support Publish/Subscriber models.

Overall, we believe that mobile crowdsensing is now achieving its maturity, being a widespread adoption of crowdsensing solutions expectable in the next few years.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

This work was partially supported by the Ministerio de Economía y Competitividad, Programa Estatal de Investigación, Desarrollo e Innovación Orientada a los Retos de la Sociedad, Proyectos I+D+I 2014, Spain, under Grant TEC2014-52690-R, the "Universidad Laica Eloy Alfaro de Manabí-ULEAM," and the "Programa de Becas SENESCYT de la República del Ecuador."

References

[1] C. to connecting the world ITU, "Ict facts and figures, the world in 2015," http://www.itu.int/en/ITU-D/Statistics/Pages/facts/default.aspx.

[2] J. A. Burke, D. Estrin, M. Hansen et al., *Participatory Sensing*, Center for Embedded Network Sensing, 2006.

[3] A. T. Campbell, S. B. Eisenman, N. D. Lane et al., "The rise of people-centric sensing," *IEEE Internet Computing*, vol. 12, no. 4, pp. 12–21, 2008.

[4] N. D. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. T. Campbell, "A survey of mobile phone sensing," *IEEE Communications Magazine*, vol. 48, no. 9, pp. 140–150, 2010.

[5] 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 *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN '10)*, pp. 105–116, ACM Press, New York, NY, USA, April 2010.

[6] M. Shin, C. Cornelius, D. Peebles, A. Kapadia, D. Kotz, and N. Triandopoulos, "AnonySense: a system for anonymous opportunistic sensing," *Pervasive and Mobile Computing*, vol. 7, no. 1, pp. 16–30, 2011.

[7] L. Ruge, B. Altakrouri, and A. Schrader, "Sound of the city—continuous noise monitoring for a healthy city," in *Proceedings of the IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, pp. 670–675, March 2013.

[8] I. Podnar Zarko, A. Antonic, and K. Pripužic, "Publish/subscribe middleware for energy-efficient mobile crowdsensing," in *Proceedings of the ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication (UbiComp '13)*, pp. 1099–1110, Zurich, Switzerland, September 2013.

[9] K. Ali, D. Al-Yaseen, A. Ejaz, T. Javed, and H. S. Hassanein, "CrowdITS: crowdsourcing in intelligent transportation systems," in *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC '12)*, pp. 3307–3311, IEEE, Shanghai, China, April 2012.

[10] R. Szabo, K. Farkas, M. Ispany et al., "Framework for smart city applications based on participatory sensing," in *Proceedings of the 4th IEEE International Conference on Cognitive Infocommunications (CogInfoCom '13)*, pp. 295–300, Budapest, Hungary, December 2013.

[11] G. Liu, M. Iwai, Y. Tobe, and K. Sezaki, "REPSense: on-line sensor data reduction while preserving data diversity for mobile sensing," in *Proceedings of the IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob '13)*, pp. 584–591, October 2013.

[12] G. Cardone, L. Foschini, P. Bellavista et al., "Fostering participaction in smart cities: a geo-social crowdsensing platform," *IEEE Communications Magazine*, vol. 51, no. 6, pp. 112–119, 2013.

[13] A. Khan, S. K. A. Imon, and S. K. Das, "An energy efficient framework for localization and coverage in participatory urban sensing," in *Proceedings of the 39th Annual IEEE Conference on Local Computer Networks (LCN '14)*, pp. 193–201, Edmonton, Canada, September 2014.

[14] H. Lu, D. Frauendorfer, M. Rabbi et al., "StressSense: detecting stress in unconstrained acoustic environments using smartphones," in *Proceedings of the ACM Conference on Ubiquitous Computing (UbiComp '12)*, pp. 351–360, ACM, Pittsburgh, Pa, USA, September 2012.

[15] X. Hu, T. H. Chu, H. C. Chan, and V. C. Leung, "Vita: a crowdsensing-oriented mobile cyber-physical system," *IEEE Transactions on Emerging Topics in Computing*, vol. 1, no. 1, pp. 148–165, 2013.

[16] Y. Zheng, T. Liu, Y. Wang, Y. Zhu, Y. Liu, and E. Chang, "Diagnosing New York city's noises with ubiquitous data," in *Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '14)*, pp. 715–725, ACM Press, September 2014.

[17] R. Pryss, M. Reichert, B. Langguth, and W. Schlee, "Mobile crowd sensing services for tinnitus assessment, therapy, and research," in *Proceedings of the IEEE International Conference on Mobile Services (MS '15)*, vol. 32, no. 2, pp. 352–359, New York, NY, USA, June 2015.

[18] M. Wisniewski, G. Demartini, A. Malatras, and P. Cudré-Mauroux, "NoizCrowd: a crowd-based data gathering and management system for noise level data," in *Mobile Web Information Systems*, vol. 8093 of *Lecture Notes in Computer Science (including sub-series Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, pp. 172–186, Springer, 2013.

[19] A. Mehrotra, V. Pejovic, and M. Musolesi, "SenSocial: a middleware for integrating online social networks and mobile sensing data streams," in *Proceedings of the 15th International Middleware Conference (Middleware '14)*, pp. 205–216, ACM Press, Bordeaux, France, December 2014.

[20] K. Han, C. Zhang, and J. Luo, "BLISS: budget Limited robust crowdsensing through online learning," in *Proceedings of the 11th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON '14)*, pp. 555–563, July 2014.

[21] V. Pankratius, F. Lind, A. Coster, P. Erickson, and J. Semeter, "Mobile crowd sensing in space weather monitoring: the mahali

project," *IEEE Communications Magazine*, vol. 52, no. 8, pp. 22–28, 2014.

[22] M. F. Bulut, M. Demirbas, and H. Ferhatosmanoglu, "LineKing: coffee shop wait-time monitoring using smartphones," *IEEE Transactions on Mobile Computing*, vol. 14, no. 10, pp. 2045–2058, 2014.

[23] X. Wu, P. Yang, S. Tang, X. Zheng, and Y. Xiong, "Privacy preserving RSS map generation for a crowdsensing network," *IEEE Wireless Communications*, vol. 22, no. 4, pp. 42–48, 2015.

[24] T. Das, P. Mohan, V. N. Padmanabhan, R. Ramjee, and A. Sharma, "PRISM: platform for remote sensing using smartphones," in *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services (MobiSys '10)*, p. 63, ACM Press, San Francisco, Calif, USA, June 2010.

[25] M.-R. Ra, B. Liu, T. F. La Porta, and R. Govindan, "Medusa: a programming framework for crowd-sensing applications," in *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services (MobiSys '12)*, pp. 337–350, ACM, Ambleside, UK, June 2012.

[26] F. Ye, R. Ganti, R. Dimaghani, K. Grueneberg, and S. Calo, "MECA: mobile edge capture and analysis middleware for social sensing applications," in *Proceedings of the 21st Annual Conference on World Wide Web (WWW '12)*, pp. 699–702, ACM Press, Lyon, France, April 2012.

[27] A. Zaslavsky, P. P. Jayaraman, and S. Krishnaswamy, "ShareLikesCrowd: mobile analytics for participatory sensing and crowdsourcing applications," in *Proceedings of the IEEE 29th International Conference on Data Engineering Workshops (ICDEW '13)*, pp. 128–135, April 2013.

[28] C. Perera, P. P. Jayaraman, A. Zaslavsky, D. Georgakopoulos, and P. Christen, "MOSDEN: an internet of things middleware for resource constrained mobile devices," in *Proceedings of the 47th Hawaii International Conference on System Sciences (HICSS '14)*, pp. 1053–1062, IEEE, Waikoloa, Hawaii, USA, January 2014.

[29] R. K. Ganti, F. Ye, and H. Lei, "Mobile crowdsensing: current state and future challenges," *IEEE Communications Magazine*, vol. 49, no. 11, pp. 32–39, 2011.

[30] W. Z. Khan, Y. Xiang, M. Y. Aalsalem, and Q. Arshad, "Mobile phone sensing systems: a survey," *IEEE Communications Surveys and Tutorials*, vol. 15, no. 1, pp. 402–427, 2013.

[31] D. Zhang, L. Wang, H. Xiong, and B. Guo, "4W1H in mobile crowd sensing," *IEEE Communications Magazine*, vol. 52, no. 8, pp. 42–48, 2014.

[32] X. Zhang, Z. Yang, W. Sun et al., "Incentives for mobile crowd sensing: a survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 54–67, 2016.

[33] L. G. Jaimes, I. J. Vergara-Laurens, and A. Raij, "A survey of incentive techniques for mobile crowd sensing," *IEEE Internet of Things Journal*, vol. 2, no. 5, pp. 370–380, 2015.

[34] H. Jin, L. Su, D. Chen, K. Nahrstedt, and J. Xu, "Quality of information aware incentive mechanisms for mobile crowd sensing systems," in *Proceedings of the 16th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc '15)*, pp. 167–176, ACM, Hangzhou, China, June 2015.

[35] J. Wang, J. Tang, D. Yang, E. Wang, and G. Xue, "Quality-aware and fine-grained incentive mechanisms for mobile crowdsensing," in *Proceedings of the IEEE 36th International Conference on Distributed Computing Systems (ICDCS '16)*, pp. 354–363, IEEE, Nara, Japan, June 2016.

[36] H. Jin, L. Su, H. Xiao, and K. Nahrstedt, "Inception: incentivizing privacy-preserving data aggregation for mobile crowd sensing systems," in *Proceedings of the 17th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pp. 341–350, ACM, Paderborn, Germany, July 2016.

[37] B. Guo, Z. Wang, Z. Yu et al., "Mobile crowd sensing and computing," *ACM Computing Surveys*, vol. 48, no. 1, article 7, 2015.

[38] F. Calabrese, L. Ferrari, and V. D. Blondel, "Urban sensing using mobile phone network data: a survey of research," *ACM Computing Surveys*, vol. 47, no. 2, Article ID 2655691, pp. 1–20, 2014.

[39] M. Talasila, R. Curtmola, and C. Borcea, "Alien vs. Mobile user game: fast and efficient area coverage in crowdsensing," in *Proceedings of the 6th International Conference on Mobile Computing, Applications and Services (MobiCASE '14)*, pp. 65–74, November 2014.

[40] E. Kwan and J. R. Getta, *Design and implementation of data stream processing applications [Dissertation]*, vol. 4611, pp. 1–142, 2010.

[41] AWS — Elastic compute cloud (EC2), https://aws.amazon.com/ec2/.

[42] Google Cloud Platform, https://cloud.google.com/.

[43] G. Merlino, S. Arkoulis, S. Distefano, C. Papagianni, A. Puliafito, and S. Papavassiliou, "Mobile crowdsensing as a service: a platform for applications on top of sensing Clouds," *Future Generation Computer Systems*, vol. 56, pp. 623–639, 2016.

[44] X. Sheng, J. Tang, X. Xiao, and G. Xue, "Sensing as a service: challenges, solutions and future directions," *IEEE Sensors Journal*, vol. 13, no. 10, pp. 3733–3741, 2013.

[45] M. Katsomallos and S. Lalis, "EasyHarvest: supporting the deployment and management of sensing applications on smartphones," in *Proceedings of the IEEE International Conference on Pervasive Computing and Communication Workshops (PERCOM WORKSHOPS '14)*, pp. 80–85, March 2014.

[46] S. Cherrier, I. Salhi, Y. M. Ghamri-Doudane, S. Lohier, and P. Valembois, "BeC 3: behaviour crowd centric composition for IoT applications," *Mobile Networks and Applications*, vol. 19, no. 1, pp. 18–32, 2014.

[47] X. Hu, X. Li, E. C.-H. Ngai, V. C. M. Leung, and P. Kruchten, "Multidimensional context-aware social network architecture for mobile crowdsensing," *IEEE Communications Magazine*, vol. 52, no. 6, pp. 78–87, 2014.

[48] A. Antonic, K. Roankovic, M. Marjanovic, K. Pripuic, and I. P. Arko, "A mobile crowdsensing ecosystem enabled by a cloud-based publish/subscribe middleware," in *Proceedings of the 2nd International Conference on Future Internet of Things and Cloud (FiCloud '14)*, pp. 107–114, IEEE, Barcelona, Spain, August 2014.

[49] N. Brouwers and K. Langendoen, "Pogo, a middleware for mobile phone sensing," in *Proceedings of the 13th International Middleware Conference*, pp. 21–40, Montreal, Canada, 2012.

[50] C. Pereral, A. Zaslavsky, P. Christen, A. Salehi, and D. Georgakopoulos, "Capturing sensor data from mobile phones using global sensor network middleware," in *Proceedings of the IEEE 23rd International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC '12)*, pp. 24–29, IEEE, Sydney, Australia, September 2012.

[51] M. Talasila, R. Curtmola, and C. Borcea, "Improving location reliability in crowd sensed data with minimal efforts," in *Proceedings of the 6th Joint IFIP Wireless and Mobile Networking Conference (WMNC '13)*, pp. 1–8, April 2013.

[52] I. Carreras, D. Miorandi, A. Tamilin, E. R. Ssebaggala, and N. Conci, "Matador: Mobile task detector for context-aware

crowd-sensing campaigns," in *Proceedings of the IEEE International Conference on Pervasive Computing and Communications Workshops, PerCom Workshops*, pp. 212–217, IEEE, San Diego, Calif, USA, March 2013.

[53] V. Agarwal, N. Banerjee, D. Chakraborty, and S. Mittal, "USense—a smartphone middleware for community sensing," in *Proceedings of the 14th International Conference on Mobile Data Management (MDM '13)*, vol. 1, pp. 56–65, IEEE, Milan, Italy, June 2013.

[54] S. Sarma, N. Venkatasubramanian, and N. Dutt, "Sense-making from distributed and mobile sensing data," in *Proceedings of the 51st Annual Design Automation Conference on Design Automation Conference (DAC '14)*, pp. 1–6, ACM Press, San Francisco, Calif, USA, June 2014.

[55] X. Sheng, J. Tang, X. Xiao, and G. Xue, "Leveraging GPS-less sensing scheduling for green mobile crowd sensing," *IEEE Internet of Things Journal*, vol. 1, no. 4, pp. 328–336, 2014.

[56] L. G. Jaimes, I. Vergara-Laurens, and A. Chakeri, "SPREAD, a crowd sensing incentive mechanism to acquire better representative samples," in *Proceedings of the IEEE International Conference on Pervasive Computing and Communication Workshops (PERCOM '14)*, pp. 92–97, March 2014.

[57] H. Aly, A. Basalamah, and M. Youssef, "Map++: a crowd-sensing system for automatic map semantics identification," in *Proceedings of the 11th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON '14)*, pp. 546–554, July 2014.

[58] C. M. Angelopoulos, S. Nikoletseas, T. P. Raptis, and J. D. P. Rolim, "Characteristic utilities, join policies and efficient incentives in Mobile Crowdsensing Systems," in *Proceedings of the 7th IFIP/IEEE Wireless Days Conference (WD '14)*, pp. 1–6, Rio de Janeiro, Brazil, November 2014.

[59] D. Zhang, H. Xiong, L. Wang, and G. Chen, "CrowdRecruiter: selecting participants for piggyback crowdsensing under probabilistic coverage constraint," in *Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '14)*, pp. 703–714, ACM Press, September 2014.

[60] T. Higuchi, H. Yamaguchi, T. Higashino, and M. Takai, "A neighbor collaboration mechanism for mobile crowd sensing in opportunistic networks," in *Proceedings of the 1st IEEE International Conference on Communications (ICC '14)*, pp. 42–47, Sydney, Australia, June 2014.

[61] C. Zhang, K. P. Subbu, J. Luo, and J. Wu, "GROPING: geomagnetism and crowdsensing powered indoor navigation," *IEEE Transactions on Mobile Computing*, vol. 14, no. 2, pp. 387–400, 2015.

[62] F.-J. Wu and T. Luo, "WiFiScout: a crowdsensing WiFi advisory system with gamification-based incentive," in *Proceedings of the 11th IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS '14)*, pp. 533–534, October 2014.

[63] Z. Wang, D. Huang, H. Wu, Y. Deng, A. Aikebaier, and Y. Teranishi, "QoS-constrained sensing task assignment for mobile crowd sensing," in *Proceedings of the IEEE Global Communications Conference (GLOBECOM '14)*, pp. 311–316, Austin, Tex, USA, December 2014.

[64] D. Zhao, H. Ma, S. Tang, and X.-Y. Li, "COUPON: a cooperative framework for building sensing maps in mobile opportunistic networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 2, pp. 392–402, 2015.

[65] M. H. Cheung, R. Southwell, F. Hou, and J. Huang, "Distributed time-sensitive task selection in mobile crowdsensing," in *Proceedings of the 16th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc '15)*, pp. 157–166, ACM Press, Hangzhou, China, June 2015.

[66] Y. Yao, L. T. Yang, and N. N. Xiong, "Anonymity-based privacy-preserving data reporting for participatory sensing," *IEEE Internet of Things Journal*, vol. 2, no. 5, pp. 381–390, 2015.

[67] C. Zhou, C.-K. Tham, and M. Motani, "QOATA: QoI-aware task allocation scheme for mobile crowdsensing under limited budget," in *Proceedings of the 10th IEEE International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP '15)*, pp. 1–6, Singapore, April 2015.

[68] C. Greco, M. Kieffer, and C. Adjih, "NeCoRPIA: network coding with random packet-index assignment for mobile crowdsensing," in *Proceedings of the IEEE International Conference on Communications (ICC '15)*, pp. 6338–6344, IEEE, June 2015.

[69] S. Distefano, F. Longo, and M. Scarpa, "QoS assessment of mobile crowdsensing services," *Journal of Grid Computing*, vol. 13, no. 4, pp. 629–650, 2015.

[70] B. Guo, H. Chen, Z. Yu, X. Xie, S. Huangfu, and D. Zhang, "FlierMeet: a mobile crowdsensing system for cross-space public information reposting, tagging, and sharing," *IEEE Transactions on Mobile Computing*, vol. 14, no. 10, pp. 2020–2033, 2015.

[71] S. Hu, L. Su, H. Liu, H. Wang, and T. F. Abdelzaher, "Smartroad: smartphone-based crowd sensing for traffic regulator detection and identification," *ACM Transactions on Sensor Networks*, vol. 11, no. 4, article 55, pp. 1–27, 2015.

[72] L. Xiao, J. Liu, Q. Li, and H. V. Poor, "Secure mobile crowdsensing game," in *Proceedings of the IEEE International Conference on Communications (ICC '15)*, pp. 7157–7162, June 2015.

[73] C. Song, M. Liu, and X. Dai, "Remote cloud or local crowd: communicating and sharing the crowdsensing data," in *Proceedings of the IEEE 5th International Conference on Big Data and Cloud Computing (BDCloud '15)*, pp. 293–297, Dalian, China, August 2015.

[74] S. Sathe, T. Sellis, and K. Aberer, "On crowdsensed data acquisition using multi-dimensional point processes," in *Proceedings of the 31st IEEE International Conference on Data Engineering Workshops (ICDEW '15)*, pp. 124–128, IEEE, Seoul, South Korea, April 2015.

[75] J. Mohite, Y. Karale, P. Gupta, S. Kulkarni, B. Jagyasi, and A. Zape, "RuPS: rural participatory sensing with rewarding mechanisms for crop monitoring," in *Proceedings of the 13th IEEE International Conference on Pervasive Computing and Communication (PerCom Workshops '15)*, pp. 378–383, St. Louis, Mo, USA, March 2015.

[76] L. Wang, D. Zhang, Z. Yan, H. Xiong, and B. Xie, "effSense: a novel mobile crowd-sensing framework for energy-efficient and cost-effective data uploading," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 12, pp. 1549–1563, 2015.

[77] S. Zhang, Q. Ma, T. Zhu et al., "PLP: protecting location privacy against correlation-analysis attack in crowdsensing," in *Proceedings of the 44th International Conference on Parallel Processing (ICPP '15)*, pp. 111–119, Beijing, China, September 2015.

[78] G. Bajaj and P. Singh, "Sahyog: a middleware for mobile collaborative applications," in *Proceedings of the 7th International Conference on New Technologies, Mobility and Security (NTMS '15)*, pp. 1–5, Paris, France, July 2015.

[79] P. Nguyen and K. Nahrstedt, "Context-aware crowd-sensing in opportunistic mobile social networks," in *Proceedings of*

the IEEE 12th International Conference on Mobile Ad Hoc and Sensor Systems (MASS '15), pp. 477–478, IEEE, Dallas, Tex, USA, October 2015.

[80] L. Xu, X. Hao, N. D. Lane, X. Liu, and T. Moscibroda, "More with less: lowering user burden in mobile crowdsourcing through compressive sensing," in *Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '15)*, pp. 659–670, ACM, Osaka, Japan, September 2015.

[81] L. Wang, D. Zhang, Y. Wang, C. Chen, X. Han, and A. M'hamed, "Sparse mobile crowdsensing: challenges and opportunities," *IEEE Communications Magazine*, vol. 54, no. 7, pp. 161–167, 2016.

[82] L. Wang, D. Zhang, A. Pathak et al., "CCS-TA: quality-guaranteed online task allocation in compressive crowdsensing," in *Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '15)*, pp. 683–694, ACM, Osaka, Japan, September 2015.