R2 - Assignment of sensing tasks to IoT devices: Exploitation of a Social Network of Objects

Luigi Atzori, Roberto Girau, Virginia Pilloni, Marco Uras
DIEE, University of Cagliari, CNIT UdR of Cagliari, Italy
{*l.atzori,roberto.girau,virginia.pilloni*}*@diee.unica.it, marco.uras.it@ieee.org*

***Abstract*—The Social Internet of Things (SIoT) is a novel communication paradigm according to which the objects connected to the Internet create a dynamic social network that is mostly used to implement the following processes: route information and service requests, disseminate data, and evaluate the trust level of each member of the network. In this paper, the SIoT paradigm is applied to a scenario where geolocated sensing tasks are assigned to fixed and mobile devices, providing the following major contributions. The SIoT model is adopted to find the objects that can contribute to the application by crawling the social network through the nodes profile and trust level. A new algorithm to address the resource management issue is proposed so that sensing tasks are fairly assigned to the objects in the SIoT. To this, an energy consumption profile is created per device and task, and shared among nodes of the same category through the SIoT. The resulting solution is also implemented in the SIoT-based Lysis platform. Emulations have been performed, which showed an extension of the time needed to completely deplete the battery of the first device of more than 40% with respect to alternative approaches.**

***Index Terms*—Social Internet of Things; Mobile CrowdSensing; resource allocation**

LIST OF MAIN SYMBOLS AND ACRONYMS

$D_{\mathcal{G}}$	Geofence size
$E_{i,k}$	Energy consumption for task k in node i
E_i^{res}	Residual energy for node i
$f_{i,k}$	Frequency at which node i performs task k
F_k^{MSF}	Minimum Sampling Frequency for task k
$\mathcal{G}$	Reference geofence
P_i^{drain}	Power consumption due to node i's functioning
s_i	Speed of node i
$\mathcal{X}_k$	Set of nodes that can contribute to task k
τ_i	Lifetime for node i
CS-ME	CrowdSensing Micro Engine
MCS	Mobile CrowdSensing
ME	Micro Engine
MEC	Micro Engine Controller
MSF	Minimum Sampling Frequency
PD	Physical Devices
POR	Parental Object Relationship
RWO	Real World Objects
SIoT	Social Internet of Things
SVO	Social Virtual Object

A preliminary version of this paper has been presented at the ICIN conference in Paris, 7-9 March 2017.
This work was partially supported by Italian Ministry of University and Research (MIUR), within the Smart Cities framework (Project CagliariPort2020, ID: SCN_00281 and Project Netergit, ID: PON04a200490)

I. INTRODUCTION

The ubiquitous and pervasive deployment of Internet of Things (IoT) objects is contributing to significantly increase the amount of data collected from the surrounding environment, without the need to build new infrastructures [1]. Technology embedded in everyday objects has raised the number of sensor-enhanced Internet-enabled devices that users take with them, from wearable devices to smartphones [2], from RFID to smart vehicles [3]. Data acquisition can be achieved either in a participatory way, where users actively contribute to the sensing task (e.g. taking a picture, twitting an earthquake), or in an opportunistic way, where sensing tasks are automatically performed by devices without the explicit involvement of users [4] (e.g. temperature monitoring, geolocation). The acquired data can be shared with other devices or delivered to the cloud, where it can be further aggregated to get what can be defined as crowd wisdom or collective intelligence [5][6]. Not only can sensing tasks be assigned to the connected and available objects, but actuating tasks as well. This is the case for instance of the local storage and transport (by moving devices) of data or the processing of a flow of data to detect events of interest happening locally. This could happen with the involvement of just the local wireless sensor network [7] or exploiting other additional services accessible through the Internet [8].

Within this context, one of the main issues is related to the fact that most of the involved objects are characterized by limited amounts of resources, such as energy, processing power, storage capacity, and transmission bandwidth. Therefore, resource management is of paramount importance, as it is confirmed by the high number of different approaches proposed in the literature to address this issue [9–12]. Another key issue is related to the involvement of the devices that should perform the sensing tasks, which requires a description of the available capabilities and the way to transfer this information to either the community of involved objects (for distributed approaches) or to the central controller.

To assign the tasks to the sensing devices, a typically adopted approach consists in developing platforms specifically devoted to this purpose, so that the IoT devices should register to this system for the specific objective of sending the sensed data to the central system, which then provides the client applications with the crowd view [13][14]. This centralized approach has key limitations in terms of scalability when handling increasing numbers of potential sensors and

requesting applications. Additionally, this strategy requires the objects to register to several other platforms when taking part in other applications (e.g. smart building, energy management), increasing the complexity in management of the objects participation to the different IoT applications. As the Cloud Computing has become a leading field for secure authentication, community sensing is in high demands of security solutions to provide user authentication, authorization and accounting for accessing legitimately the cloud services [15].

This paper handles the mentioned issues by leveraging the Social Internet of Things (SIoT), which is a novel communication paradigm according to which the objects connected to the Internet create a dynamic social network that is mostly used to implement the following processes: route information and service requests, disseminate data, and evaluate the trust level of each member of the network. The contributions of the paper are the following: i) we define how the SIoT social network can be used to find the objects that can contribute to the distributed sensing operations and can collaborate in the exchange of information about the energy consumption, which is important when allocating tasks to the community of nodes; ii) we present the implementation details by showing how this application runs in the Lysis platform implementing the SIoT paradigm; iii) we propose a new algorithm to address the resource management issue so that sensing tasks are fairly assigned to the objects and no node is overloaded with respect to the others. Even though the algorithm can be applied to other application scenarios, in this paper we often refer to Mobile CrowdSensing (MCS), a pervasive sensing paradigm where users' mobile objects can often replace static sensing infrastructures; iv) we illustrate the emulation-based experiments showing the performance in terms of nodes lifetime extension and sampling frequency accuracy.

The remainder of the paper is structured as follows. Section II provides an overview of the background related to SIoT, the Lysis platform and MCS past works. In Section III the approach adopted to implement the MCS in the Lysis architecture is presented. Section IV describes the algorithm for resource allocation and how the energy profile is created per device. Tests results and performance evaluation are discussed in Section V. Section VI draws final conclusions.

II. Background

In the following, we briefly describe the SIoT paradigm and the Lysis implementation. We also present the major works in MCS.

A. The Social Internet of Things

Recently, some studies have addressed the issues related to the management and effective exploitation of huge numbers of heterogeneous devices and have found a solution in the use of social networking concepts and technologies. For instance, the object ability of having conversations like humans has been introduced in [16]. In [17], explicitly, the Social IoT (SIoT) concept is formalized, which is intended as a social network where every node is an object capable of establishing social relationships with other things in an autonomous way according to rules set by the owner. The SIoT relies on some key types of relations:

- *Ownership Object Relationship* (OOR): is created between objects of the same owner
- *Co-location Object Relationship* (CLOR): is created between still devices located in the same place
- *Parental Object Relationship* (POR): is created between objects of the same model, vendor and production batch
- *Co-work Object Relationship* (CWOR): is created between objects that meet each other in the owners' workplace
- *Social Object Relationship* (SOR): is created as a consequence of frequent encountering between objects, as can happen between smartphones of students attending the same class.

These relationships are created and updated on the basis of the objects' features (such as object type, computational power, mobility capabilities, brand) and activity (frequency in meeting the other objects, mainly). Recently, some works have studied how the SIoT can be used to monitoring certain types of smartphone activities to minimize the users' concerns about accessibility and tracking in smart social spaces [18]. The resulting objects social activity can also be used for the management of the trust in the object-to-objects communications [19] [20].

B. The Lysis implementation of SIoT

The solution we proposed in this paper relies on the cloud Social IoT architecture, named Lysis [21], that foresees a four-level structure as shown in Figure 1. Its lowest layer is populated by the Real World Objects (RWO). At this layer, the Physical Devices (PD) directly access to the platform via direct links to the Internet, while other objects (more resource-constrained) need to rely on Gateways (GWs) for the Internet connection, allowing them to send data to and receive commands from the level above. PDs and GWs are able to perform basic tasks, such as secure communication with the respective virtual counterparts, as well as management and presentation of data coming from sensors. On top of this, the Virtualization layer directly interfaces with the real world and is populated by the Social Virtual Objects (SVOs). The virtualization functionalities are common to most IoT cloud-based platforms to address most of the issues related to the low level of resources the IoT objects are equipped with [22]. In Lysis, these functions are also extended with the social capabilities assigned to the agent implementing the virtual objects so that it is able to establish and manage friendships with other SVOs autonomously with respect to the owner. Accordingly, a network of digital counterparts of the physical devices is created and available at this layer. This can be used to search objects and information they produce, evaluate the trust level, create groups to foster collaboration. The Aggregation layer is responsible for composing several SVOs into entities with extended capabilities, called Micro Engines (MEs); the ME is the entity that implements part of the application logic performed at the upper layer. In each ME,

the output for a request coming from an application can be reused to serve requests from different applications that require the same information or service to save bandwidth and CPU. Finally, at the Application layer, user-oriented macro services are provided (APP).

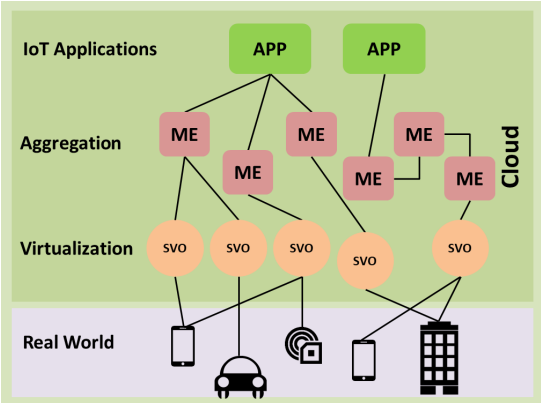


Fig. 1: The four levels of the Lysis architecture

C. Mobile CrowdSensing

In the MCS paradigm, the objective is that of performing an application, which needs large amounts of sensed data, by gathering this data from a high number of devices, provided that they are capable of fulfilling the application requirements [3]. These requirements typically include constraints about the location and time interval where the data were collected [23], which affect the accuracy of the overall measure [24].

A huge number of MCS-based applications have been proposed in the literature. Thanks to its characteristics, the MCS has proven to be particularly advantageous for, but not limited to: ambient monitoring [25][26], pattern recognition, mainly for healthcare analytics [27] or smart city and route planning applications [28], and social-related applications, where users share and compare data about themselves with a community [29][30]. Heterogeneous networks offer new opportunities for MCS technologies supporting connectivity, collaborative monitoring and data routing / transmission. Routing protocols for wireless networks cannot handle a highly dynamic topology typical of the MCS scenario. Fortunately, recent advances in the Internet of Things offer optimized routing solutions [31].

The impressive spread of smart mobile devices such as smartphones, tablets and smart watches, have contributed to the development of MCS, which, compared to traditional sensor networks, entails significant advantages, in particular with reference to deployment costs and coverage extension [4]. Indeed, since any smart mobile device is equipped with many different sensors (e.g. GPS, camera, microphone, light), any device that is in the right place at the right time and which information quality requirements are satisfactory can be used to contribute to MCS data collection. Considering that those devices are typically provided with networking functionalities (e.g. WiFi, 3G, Bluetooth), spending money and time on deploying new infrastructures to cover new areas is not needed.

Nevertheless, MCS inherited one of the major issues of traditional sensor networks: the limited amount of resources available on devices [32] (e.g. available energy, storage capacity, computing speed). Fair resource allocation mechanisms can be applied to overcome this problem. Some examples are provided by the studies presented in [9–12]. The Context-aware Mobile Sensor Data EngiNe (C-MOSDEN), a scalable energy-efficient data analytics platform for on-demand distributed mobile crowd, is proposed in [9]. The C-MOSDEN platform collects data only when they are relevant to the required context, and it is able to autonomously configure, e.g. selecting the most appropriate communication channel, or sampling rate. A similar approach is used in [10], where particular attention is put on satisfying global sensing coverage requirements and avoiding redundant sensory activities. In [33], a distributed framework for data collection is proposed to minimize the cost of sensing by estimating the sensing potential and the data collection utility. The former defines the best candidate among participants to perform sensing. The latter is computed by accounting the sensing interest of the involved applications. Energy efficiency and battery saving are the main objectives of Re-OPSEC [11], where tasks are assigned to smartphones according to their expected opportunity of succeeding in the sensing task, and their residual battery charge. In [12], the concept of social vector, where node attributes are defined, is introduced. Social vectors are used to find the most accurate matching between a task with its requirements and a node with its resources. However, none of the analyzed strategies (summarized in Table I) exploits social relations among objects to improve the system performance.

TABLE I: Major approaches for MCS

Ref.	Energy Awareness	Sensing service description
Ruge et al. [25]	No	ambient monitoring
Predić et al. [26]	No	activity based participation
Lathia et al. [27]	No	context-based
Goldman et al. [28]	No	smart city
Wang et al. [29]	No	community based
Eisenman et al. [30]	No	social vectors based
Perera et al. [9]	Yes	location and activity-aware
Skorin-Kapov et al. [10]	Yes	cloud-based quality-driven management
Bradai et al. [11]	Yes	expected opportunity based
Hu et al. [12]	Partially	application-oriented service collaboration
Capponi et al. [33]	Partially	based on sensing potential and data collection utility

III. Exploiting the Social IoT for Sensing Task Assignment in IoT

This Section presents our solution which exploits the SIoT paradigm to assign sensing task to IoT devices and relies on the Lysis platform. As stated in the Introduction, we refer to this objective of sensing task assignment as an important function of the MCS, which is often cited as a reference scenario for our treatment. However, the proposed solution can be also applied to other application scenarios characterized by geolocated sensing tasks that need to be performed by a conspicuous number of devices.

A. Preliminaries

As shown in the previous Section, in Lysis the socialization algorithms are implemented in the second level where the resulting social relations are created and managed. All the applications running in Lysis pass through the MEs, which represent simple processes that exploit the information and the services provided by the SVOs. Clearly, in our view, all the mobile devices that take part in the MCS operations need to be registered to the platform and need to have their own SVO running. The MCS is one of the MEs that can be used by the applications running in the IoT platform.

Any application can then exploit the SIoT social graph for the major tasks that are needed when a huge and heterogeneous community of objects is involved: find the potential provider of information and services, evaluate the trust level of each object, and disseminate information, just to cite the major ones. As far as the MCS operation is concerned, the search of potential sources of data is the operation of major interest. Indeed, to find the required resources each application sends a query with the semantic description of the needed resources to the ME Controller (MEC), as shown in Figure 2. The MEC is in charge of forwarding the query to a special SVO called SVO-Root (SVO-R) which forwards the query to its friends and friends of friends in order to get a reply with a list of resources and their access keys. This happens also to the MCS ME, which is inquired with the tags that describe the information that each node should be able to provide, e.g., the physical magnitude of interest, the geographical position, the sensing accuracy.

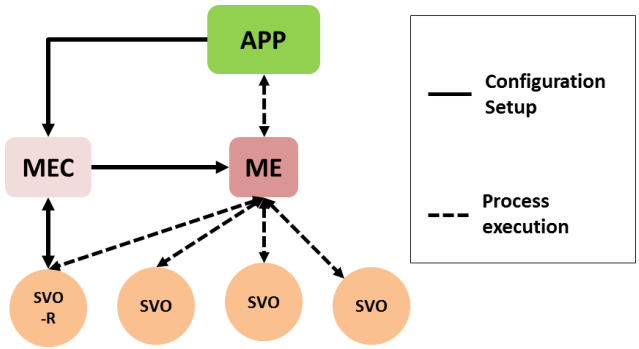


Fig. 2: Configuration chain

After having found the required SVOs, the MEC links them in the ME devoted to the MCS, which then handles the SVOs for the processing needed by the requiring application (those requiring MCS services). For our specific applications, this processing means inquiring about the selected nodes on the basis of the algorithm described in the following Section and compute the crowd view. The ME is able to configure each SVO to either perform some operations in the background or to trigger their resources so as to provide an active participation in the required crowdsensing process.

B. The CrowdSensing Micro-Engine

We refer to the ME in charge of implementing the crowd-sensing algorithm as CrowdSensing ME (CS-ME). This is the one that provides the applications running in the Lysis platform with the services to get the view from objects mobile crowd about a given physical magnitude. These services are accessible through different queries, that are handled by the MEC in JSON format and that are passed to the CS-ME. The most important query allows the application to specify the maximum number of requested resources (*limit*), the depth of search in the social graph (*hop*), the coordinates of the geofence where the search should be performed, a description of the resources in text format in the *description* parameter, and the type of relationship which can be exploited by the SVO search process is specified in the *relationship* field, as shown in Listing 1. The query also includes different parameters to control the amount of resources for a given crowd sensing task or the required accuracy. In the example of Listing 1, the minimum sampling frequency (MSF) is used, which is the minimum number of samples per second to be provided by the crowd for this task. Note that this parameter is also the one we consider in the following of the analysis. From the Figure, you can note also the owner-key field, which is the one that identifies the user that has generated the query and somehow drives determine the amount and which resources the application is entitled to access to.

```
{"owner-key":"ovdsljvdsoibewdp",
 "owner-id": "user@gmail.com",
  "limit": "20",
  "hops": "2",
 "longitude":"9.109900",
 "latitude":"39.229055",
 "range":"30",
 "MSF":"0.4"
 "description":"sensor brightness
    temperature pressure",
 "relationship":[{"type":"SOR"}]
}
```

Listing 1: Textual description of the resources requested by an application

It may happen that the same query was already generated by another application, which would mean that another instance of the CS-ME would be running to do the same job. No to waste resources, this event is checked so as the same instance is used to satisfy two (or more) applications with the same MCS need. If this is the case, the MEC says to the CS-ME to add the new query ID to the previous one(s) to respond to the new request with the data produced for the previous requests. Clearly, the CS-ME has to check if the input resources and the geofence are the same as the previous requests. If so, the MSF is set to the highest requested value, while the output of the crowdsensing process is returned also to the applications with less stringent requirements.

If it is not the case and a new instance needs to be created, the CS-ME executes the SVO search process to find the needed resources. The SVO search process then sends to the CS-ME the reference to the found resources. Listing 2 shows an excerpt of the JSON configuration sent to the CS-ME after the search process terminates. For each resource,

access permissions (*Permission*), the API key (*Key*), the URL of the SVO (*ResourceURL*) and the SVO's sensor name (*FeatureName*) are indicated. In the excerpt in Listing 2, the first resource is public and does not require any access key, whereas the second SVO can share its resource only with friends and therefore it asks to specify the API key in the JSON configuration file.

```
{"input-id":"1",
 "resources": [
 {
  "Permission":"public",
  "Key":"",
  "ResourceURL":"svotest100.appspot.com",
  "FeatureName":"BRIGHTNESS"
 },
 {
  "Permission":"friend",
  "Key":"thisisthe2ndfriendkey",
  "ResourceURL":"svotest101.appspot.com",
  "FeatureName":"TEMPERATURE"
 }]
}
```

Listing 2: The list of SVOs which can participate in the CrowdSensing process is sent by the MEC to the CS-ME

All the identified SVOs are asked by the CS-ME to provide the power-profiles to all the participating SVOs. These are also asked to alert the CS-ME whenever they exit or enter the geofence of interest for this task. Since it is an event-driven process, the CS-ME waits for any alerts from SVOs to assign frequencies to the available SVOs according to the algorithm described in Section IV and shown in Algorithm showed in Figure 3. Accordingly, this ME continuously communicates with the SVOs to re-assign tasks to satisfy the application requests, according to the mobility of the mobile users and their energy. It is also worth noting that the CS-ME has the ability of self-healing, so that if a malfunction occurs in one or more of the resources, it can ask to the MEC for a reallocation of resources.

C. Advantages provided by SIoT

The SIoT paradigm relies on the construction of a peer-to-peer social network which is built on the basis of inter-objects interactions and similitudes. Such a network is exploited in major phases of the MCS activities bringing major advantages:

- Search of nodes to be involved: each node advertises its own functionalities that are then made available on the SIoT social network for the benefits of the other members. As explained in Section III-B, this is exploited by the CS-ME to build the group of objects with the required capabilities. As it is highlighted in the example of Listing 1, the type of relationship is also used to search for the objects that are useful for the crowdsensing application. Each object (according to the rules set by the owner) can decide to make each functionality private (available only to the owner), friend (accessible only to the friends) or public (anybody can access to). Whereas all the relationships types can be used, the SOR and the CLOR are the most useful.
- Trustworthiness evaluation: the trustworthiness of the nodes is evaluated by the peers in the social network, as it is explained in [20]. This is a functionality that is made available for all the applications deployed on top of the SIoT network and it is quite useful for the selection of the nodes to be involved in the MCS applications. For this feature, all the relationship types are used, with different weights.
- Sharing of the device energy profile: as it will be made clearer in the following, there is the need to create an energy consumption profile per node involved. To avoid repeating the same computation in each node, once it is created by a node the profile is shared with all the others with similar characteristics exploiting the POR relationships (i.e., those with similar characteristics).
- Update of nodes status: the objects social network is also exploited to exchange the changing context information among groups of friends. The speed of the friend nodes and the relevant battery status represent information of interest for the MCS nodes and would make faster the selection of which nodes to involve according to the changing context. CLOR and SOR relationships are used in this case.

Whereas some of these features are utilized in the current proposal, some others have not been included in the implementation as the number of nodes in the running Lysis platform is limited and would not allow for an efficient evaluation of the relevant benefits.

IV. Energy-based Source Selection

In this Section, we first describe the algorithm that has been devised to select the crowd members for the required sensing tasks and then we describe how we generate the device energy profile.

A. The Resource Allocation Algorithm

The resource allocation strategy used in this paper relies on an optimization algorithm where the ME decides the amount of resources to allocate to a task, according to the requirements coming from the higher layers and to the available resources. This strategy starts by choosing which of the nodes inside or close to a geofence can collaborate to get the answer to the pending query. Then, the optimization algorithm described in Figure 3 allocates the resources made available by the assigned nodes, so that the task is performed in a way that affects the system lifetime as little as possible.

To correctly select the set of nodes $\mathcal{X}_k$ that can contribute to task k, the ME considers the ones that are less likely to cause frequent reconfigurations of the system, among those that are inside or close to the reference geofence. Starting from this consideration, two features are considered to exclude nodes from set $\mathcal{X}_k$: I) residual battery charge that is considerably

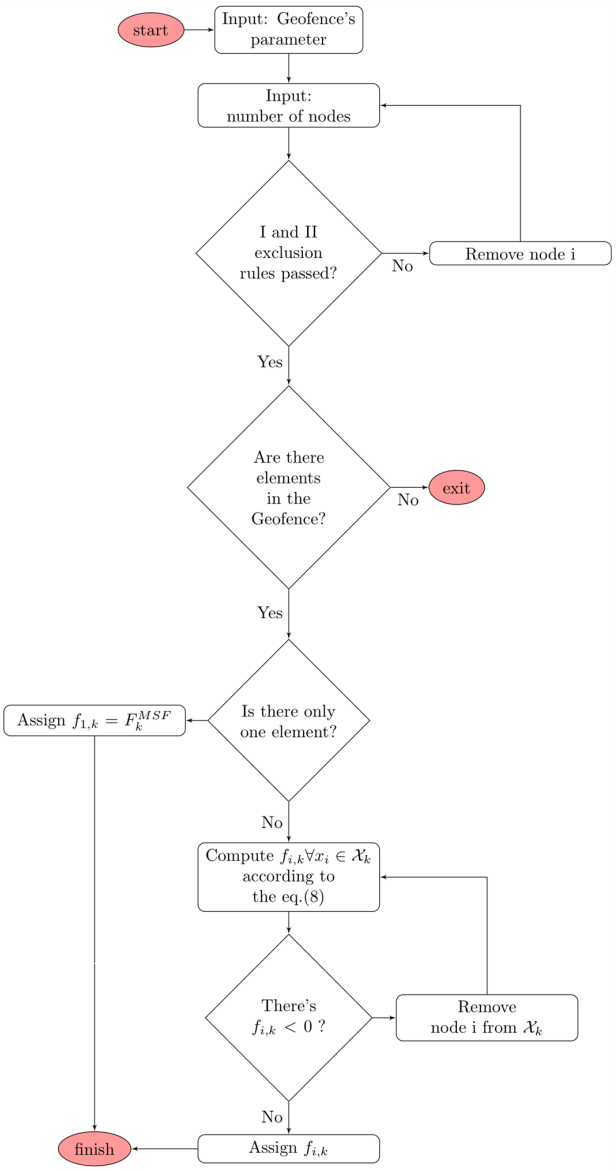


Fig. 3: Flow chart of the proposed task allocation algorithm

lower than that of the other nodes; II) speed that is too high, and that would quickly take the node out of the geofence.

With respect to the *I exclusion rule*, node i is not included in the resource allocation process if its residual energy is relatively too low, i.e. if the following inequality holds for its residual energy E_i^{res}

$$E_i^{res} \leq \alpha \cdot \bar{E}^{res} \tag{1}$$

where $0 < \alpha << 1$ is a parameter that sets the relative threshold for E_i^{res}, and $\bar{E}^{res}$ is the average residual energy for the considered set of nodes.

With reference to the *II exclusion rule*, node i is considered to be too fast for the geofence $\mathcal{G}$ under evaluation if its speed s_i is too high with respect to the geofence size $D_{\mathcal{G}}$ and to the MSF frequency F_k^{MSF} for MCS task k, so that the exclusion rule is defined as follows

$$s_i \geq \beta \cdot D_{\mathcal{G}} \cdot F_k^{MSF} \tag{2}$$

where $0 < \beta \leq 1$ is a parameter that sets the relative threshold for s_i. Note that when $\beta = 1$, the threshold is set so that the node is expected to be able to provide one sample when crossing the geofence if it was asked to sample at frequency F_k^{MSF}. As the sensing task is shared with other nodes, this parameter is set lower than 1. The impact on setting the two parameters α and β is discussed in the experiments Section.

The nodes remaining in the set $\mathcal{X}_k$ are the nodes that pass the exclusion rules. Accordingly, the ME optimally allocates the resources made available by the nodes in $\mathcal{X}_k$. The resource allocation algorithm proposed in the following relies on the principle used also in the algorithm proposed in [34], with consists in setting the nodes sensing frequency so that the resulting lifetime is equal in all the members. However, herein the assignment of the frequencies is done in the platform by the CS-ME which has the information about the status of all the nodes involved. Differently, in the cited article the allocation algorithm is distributed, and the nodes have to adjust the sensing frequency according to the available information about the status of the other nodes, which is exchanged making use of a consensus-based algorithm. Additionally, in [34] different types of resources are considered when deciding for the frequency to assign to each node, i.e., lifetime, object lifetime, storage capacity, and processor and data throughput. Differently, in the following, we focus on the energy consumption caused by the sensing activity and the transmission of the data at the assigned frequency.

As done in other papers (such as in [35]), we consider the system's lifetime τ as the time until the first node belonging to that system depletes its residual energy

$$\tau = \min_i \tau_i \tag{3}$$

where τ_i is node i's lifetime, defined as

$$\tau_i = \frac{E_i^{res}}{P_i^{drain} + \sum_k E_{i,k} f_{i,k}} \tag{4}$$

with: P_i^{drain} power consumption due to other activities of the nodes that are not assigned by the ME (e.g. other tasks that are started directly by the user); $E_{i,k}$ is the energy consumption needed by node i to perform task k, computed; $f_{i,k}$ is the frequency at which node i performs task k. Equation (4) entails that the ME is able to affect the lifetime of the system made of the nodes of a geofence $\mathcal{G}$, by appropriately adjusting the frequency at which the involved nodes perform the required tasks.

The objective of the framework proposed in this paper is that of assigning tasks so that there is not a node that depletes its available resources earlier than the others. When addressing lifetime, this condition is satisfied when the workload is fairly distributed among all the nodes, i.e. when all the nodes tend to the same lifetime. In other words, taken any two nodes $\{i, j\} \in \mathcal{G}$, their workload is fairly distributed if, after the task assignment, they have the same lifetime $\tau_i = \tau_j = \tau, \forall \{i, j\} \in \mathcal{G}$. Therefore

$$\sum_k \gamma_{i,k} f_{i,k} + \frac{P_i^{drain}}{E_i^{res}} = \sum_k \gamma_{j,k} f_{j,k} + \frac{P_j^{drain}}{E_j^{res}} \tag{5}$$

where $\gamma_{j,k} = E_{i,k}/E_i^{res}$. We define the total amount of power consumption contributions of node i with the exception of task k as $\delta_{i,k} = \sum_{l \neq k} \gamma_{i,l} f_{i,l} + P_i^{drain}/E_i^{res}$. Substituting it in Equation (5)

$$f_{j,k} = \frac{\gamma_{i,k}}{\gamma_{j,k}} \cdot f_{i,k} + \frac{\delta_{i,k} - \delta_{j,k}}{\gamma_{j,k}} \tag{6}$$

To satisfy accuracy requirements coming from the application layer, the following relation about the F_k^{MSF} has to be fulfilled

$$F_k^{MSF} = \sum_j f_{j,k} = \sum_j \frac{\gamma_{i,k}}{\gamma_{j,k}} \cdot f_{i,k} + \sum_j \frac{\delta_{i,k} - \delta_{j,k}}{\gamma_{j,k}} \tag{7}$$

Therefore, the ME can assign a frequency to each SVO i for each task k, according to the following Equation

$$f_{i,k} = \frac{1}{\gamma_{i,k}} \cdot \left(\sum_j \frac{1}{\gamma_{j,k}} \right)^{-1} \cdot \left(F_k^{MSF} + \sum_j \frac{\delta_{j,k}}{\gamma_{j,k}} \right) - \frac{\delta_{i,k}}{\gamma_{i,k}} \tag{8}$$

It may happen that the result of Equation (8) is negative: it is the case of a node whose lifetime is so low it cannot take any other load because its lifetime is already lower than that of the others even assigning the whole task k to the other nodes. Since a negative frequency does not have physical meaning, in this case frequency $f_{i,k}$ is set to 0, and Equation (8) is evaluated again for every node, excluding node i from the computation.

In the following, we provide the pseudocode for Algorithm 1.

Algorithm 1 MCS task allocation algorithm

Require: $|\mathcal{X}_k| > 0$
Inputs: $\mathcal{X}_k$, F_k^{MSF}, α, β, $\bar{E}^{res}$, $D_{\mathcal{G}}$
while alert **do**
  Delete from $\mathcal{X}_k$ the nodes that don't pass the *I exclusion rule* (Eq.(1)) $E_i^{res} \leq \alpha \cdot \bar{E}^{res}$
  Delete from $\mathcal{X}_k$ the nodes that don't pass the *II exclusion rule* (Eq.(2)) $s_i \geq \beta \cdot D_{\mathcal{G}} \cdot F_k^{MSF}$
  if $|\mathcal{X}_k| > 0$ **then**
    if $|\mathcal{X}_k| = 1$ **then** $f_{1,k} = F_k^{MSF}$
    else
      for all $i \in \mathcal{X}_k$ **do**
        Set $f_{i,k} = \Gamma \cdot F_k^{MSF} + \Phi$ (Eq. (8))
        if $f_{i,k} < 0$ **then**
          Set $f_{i,k} = 0$ and remove i from set $\mathcal{X}_k$
          Repeat from step 4
        end if
      end for
    end if
  end if
  alert = false
end while

B. Generation of the Device Energy Profile

For the implementation of the described algorithm, for each device type i and task k, we need to know the required energy $E_{i,k}$. The last is made of two independent tasks: the sampling, which requires an access to the physical sensor through the system drivers; and data transmission, which is characterized by the data connection configuration (mostly data rate and connection technology). These two independent components can be modeled with energetic consumption status diagrams, both for the sensors as presented in [36] and the radio interface referring to Radio Resource Control (RRC) protocol [37][38] as follows

$$E_{i,k} = E_{i,k}^{sens} + E_{i,k}^{tx}. \tag{9}$$

Each term on the right side in Equation (9) (E^x, with $x = sens, tx$) is defined as follows [39] (we omit the subscript i and k for presentation convenience)

$$E^x = \sum_s E_s^x + \sum_s \sum_{h, h \neq s} E_{s,h}^x \times C_{s,h}^x, \tag{10}$$

where: observation and modeling time is divided in intervals during which the status of the considered component (CPU, antenna and sensor) does not change and s indexes these periods; E_s^x represents the energy during each of these periods, $E_{s,h}^x$ is the energy to transfer from state s to state h; and $C_{s,h}^x$ counts the numbers of passages from state s to state h. To better explain these aspects, in Figure 4 we show an example of the energy required to perform a single sampling of the brightness sensor and the transmission of the acquired value. In this Figure, two phases have been detected, which are highlighted in different colors. During each phase, we have a different state for the sensor and the antenna and the relevant energy components need to be considered, as expressed in Equation (9). Note the different energy consumption rates that characterize the two phases. This specifically refers to a Motorola MotoG2015 when transmitting using the UMTS radio interface. Accordingly, there is the need for a device energy profile which is meant to be used to estimate the energy consumption varying the activity performed by the device and the data connection technology used.

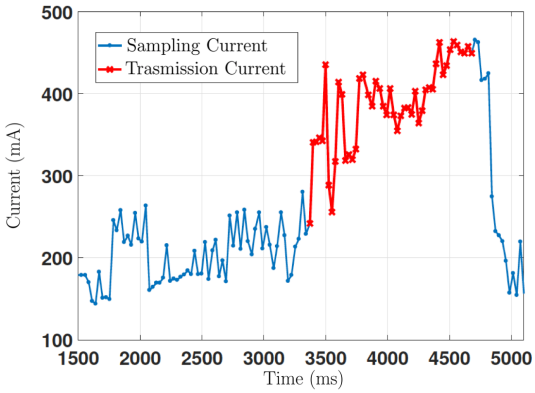


Fig. 4: Battery drain before (while sampling the brightness sensor) and during transmission

According to the previous considerations, we have built an experimental based model which provides the energy consumption per joint sensing and transmission tasks on the basis of the transmission technology and RSSI (Received Signal Strength Indicator) value. Specifically, we have built

TABLE II: Actions performed in the states

State	Action
x_1	Idle
x_2	Base current estimation
x_3	Transmission and sampling execution
x_4	Computing task cost
x_5	Send value to SVO

TABLE III: Events that trigger a state transition

Edge	Event
e_1	Already computed technology and RSSI level
e_2	New technology and RSSI level detected
e_3	Repeat N times
e_4	End of transmissions pool

a lookup table which is used to identify the required energy: $\{RSSI_j, T_j^{Tech}, D_j^{Model}\} \rightarrow E_{j,k}$. In this notation T_j^{Tech} indicates the transmission technology that, in our experiments, we considered as the followings: UMTS, LTE and Wi-Fi. D_j^{Model} identifies the specific device model. Additionally, in line with the previous consideration, the type of task k does not influence the look-up process.

To model the dependence on the data transmission technology, we relied on HTTPs transmissions from the device under observation with a payload of 100KB under different received signal conditions and different technologies (WiFi, UMTS and LTE). For each technology and for each received power level, the device performs a series of N HTTPs transmissions while the battery discharge current is monitored by means of the built-in API of the Android operating system. We have found that $N = 30$ was experimentally sufficient for an acceptable accuracy level. Sasu Tarkoma et al. in [40] show different approaches to estimate the energy consumption. The methodology adopted in this work does not require external measurements or calibrations but only relies on the modeling software that runs in the device. However, it has the drawback that the measurements on the device introduce systematic errors due to the overhead required to carry out the measurement itself. Whereas the transmission and sampling energy consumptions are estimated separately, the two operations are performed one after the other when performing the N tests.

The steps of the power profile learning process are described in the diagram in Figure 5. To support the explanation of the process, Tables II and III describe the process states and the events triggering the transitions, respectively. The creation of a new energy profile starts when required by the SVO once it finds that it is not available among those already created on the basis of the required transmission technology and RSSI level. The first task performed (state x_2) is the estimation of the *base* current (i.e., the current absorbed when no sampling and transmissions are performed), which is the average amount of current delivered by the battery before the transmission starts. This is considered as an offset to be subtracted from the average value during transmission. Then, the already mentioned N HTTP calls to the SVO are executed (typically $N = 30$). The estimated energy needed per each sampling and transmission task $\overline{E}$ is computed as follows

$$\overline{E} = \frac{1}{N} \sum_{r=1}^{N} E_r \tag{11}$$

with

$$E_r = (\overline{i_r} - i_{base}) V \Delta t_r \tag{12}$$

where E_r represents the energy consumption needed by the node to perform the r-th HTTP test call to the SVO and depends on the observed average current $\overline{i_r}$ minus the physiological absorption current i_{base} and the needed time Δt_j. Whereas the sensing and transmission procedures are executed one after the other for each repetition, the estimation of energy consumption is computed differently for these two types of activities.

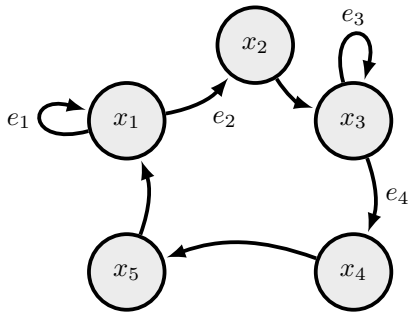


Fig. 5: State diagram of energy profile learning process

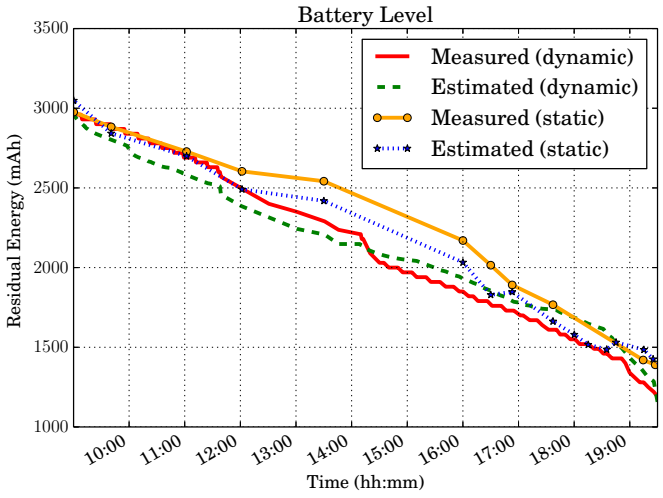


Fig. 6: Battery level: comparison of the measured values with the estimated ones, for the static and dynamic cases

To evaluate the accuracy of the model, we compare the model-based estimation with respect to the real measurement. Figure 6 shows the estimated and measured values of residual battery energy when the smartphone was either static or moving, during one experimental run. We can see that the estimation provides better results in the static case, as it was

TABLE IV: Average error for dynamic and static model

Brand and model	Mean Error (Static Model - Dynamic Model)
Asus Zenfone 2	1.12% - 3.57%
Honor 9	3.74% - 6.95%
Xiaomi Redmi 4 Pro	3.81% - 6.89%
Huawei P20 Lite	4.86% - 5.76%
Samsung Galaxy S6	2.37% - 4.52%
Motorola MotoG 2015	4.59% - 6.54%

expected as fewer changes in the used transmission technologies happen. In fact, in the dynamic case, the smartphone was moving around the university campus, which is partially served by the Eduroam WiFi network and clearly in any place by the cellular data service, so that horizontal and vertical handovers happened. This has caused changes in the energy consumption trend that were more difficult to follow than in the static case, where no handover events occurred. Despite the different state of mobility, the slopes of the two curves are similar, as a whole. This is due to the fact that on average the energy consumption for the different technologies resulted to be equivalent. Additionally, the major causes of battery discharge are due to the state of display, CPU workload, radio interfaces and location services and the human-battery interaction [41, 42] in both cases were similar. Finally, the average error observed over 100 runs was of 4.59% and 6.54% for the static and moving cases, respectively.

To investigate on the errors that could be introduced with different device types, we performed the same test with other smartphones whose results are shown in Table IV (recall that the results in Figure 6 refer to the Motorola MotoG 2015 device). It resulted that the average errors for each device, both for the dynamic and static cases, are very similar among the different devices and always below 7%. This was expected as the profiles have been created specifically for each tested device. The limited differences in the average error are due to the variability in energy consumption at the same device status (in terms of communication technology and RSSI) which is unpredictable and characteristic of each device model.

This learning process takes advantage of the SIoT network to avoid repeating it when not needed so that the SVOs exploit the social relationships among devices of the same type. Specifically, the POR established among objects of the same model and brand is the one that is used for this purpose. Note that the energy profile needs to specifically match the physical device model. In case the needed profile is found, the smartphone just registered on the Lysis platform does not have to start the energy profile learning process but borrows it from its friends. If the power profile of friends is only partially completed (e.g. a given transmission technology has not been analyzed yet), the SVO sets the learning procedure on the smartphone for the missing information. When a new profile is built or finalized, it is forwarded to the first circle of friends following the POR friendship. In this way, the power profile learning effort is shared among the devices of the same type.

V. Performance Analysis

To evaluate the performance of the devised solution we have performed several emulations, which are aimed to show how the proposed strategy is able to achieve the objective of avoiding the depletion of the device battery. The results are referred to MCS, which is one of the possible application scenarios for the proposed algorithm. We first describe the testbed setup then we present the results.

A. Testbed setup

In crowdsensing, finding a good number of participants available to perform different runs for long-time periods to perform tests has always been a problem and the most adopted solution is the exploitation of simulators for both applications and devices [43]. For this reason, we decided to develop an SVO emulator that leverages the observation of real devices in terms of mobility, energy consumption and performed activities. Accordingly, we first acquired data from 20 *real* Android devices owned by students and researchers, who spent most of the day at the University campus. The Android app and the MCS device driver used in this test are available on the Lysis platform and require Android 5.0 or higher. By observing data from real devices, we created the energy profiles for 6 different Android smartphone models and mobility traces of people moving within the University campus.

In our prototype, the CS-ME is implemented and run on the real Lysis platform, which communicates with the emulated SVOs. The SVOs select the energy cost on the basis of the used transmission technology and sensor using the created set of energy profiles from the real devices. From the observation of several typical days at the university campus where classes in different rooms were held, we have been able to isolate recurring patterns in the habits of students and researchers. We used these patterns to build synthetic mobility traces to use in our emulator (which included also the transmission technology selected by the smartphones for each position).

In our emulations, the user path during the day is built by composing parts of real patterns observed from the real users. By combining randomly these patterns we could perform different runs and extend the sessions temporally. Specifically, the initial points in the patterns were modified geographically and temporally randomly in a range of $(0, 10)$ meters and of $(0, 10)$ minutes, respectively. The battery drain was modeled through a parametric function from the observed data of real smartphone behavior. It depends on user usage, sampling frequency and quality of the connection. The resulting emulator exposes the same API of Lysis SVOs so that we are able to test a real CS-ME to evaluate frequency error, latency in the setup phase and capacity of the algorithm to quickly follow the changes in the crowd availability.

Table V summarizes the initial setting of the major nodes' parameters as it arises from the described analysis of the real devices. These are distributed in three different geofences: the Lidia building where classes are held, the MCLab laboratory and the park, all located within the same Engineering campus.

B. Analysis of Results

To evaluate the performance of the algorithm, we executed each run until all the devices ran out of battery. The case

TABLE V: Initial setting of major nodes' parameters

Name	Range
E_i^{res}	(3000,4200) mAh
P_i^{drain}	(150,450) mW
$E_{i,k}$	(0.0095,0.035) mWh
s_i	(0.5,10) m/s

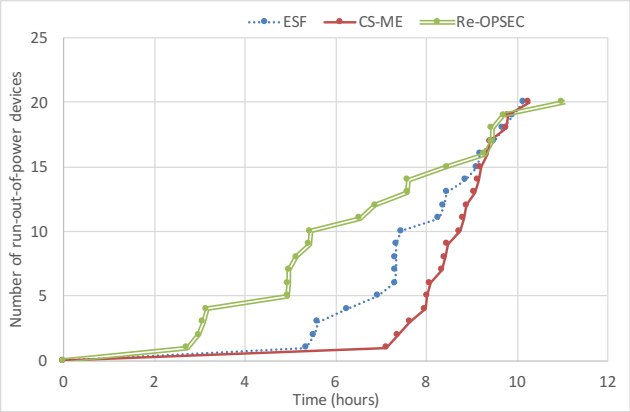


Fig. 7: Comparison of the CS-ME algorithm with an equivalent scenario where a constant sampling frequency is assigned to each device taking part in the crowdsensing and with the Re-OPSEC algorithm [11]

we consider here is the one where MSF was set to 0.067 Hz, which is compared with the case of crowdsourcing with an equally set sampling frequency (ESF) of $0.067/20$ Hz for each node (the number of considered devices was constant and equal to 20, all with the battery fully charged). We further compared the algorithm with the Re-OPSEC algorithm proposed in [11]. According to this approach, a single sensing task is assigned to the most convenient sensing nodes at each time period, i.e. the inverse of the MSF value. Convenience is evaluated according to the difference between the opportunity and the cost to perform a given task. In our case, the opportunity of each node i to perform task k is equal to 1 whenever the node is inside the geofence, equal to 0 otherwise. The cost is equivalent to the decrease in lifetime required to perform task k, i.e. $(E_i^{res} - E_{i,k})/E_i^{res}$. Figure 7 shows the number of devices that ran out of battery for the two cases. It is evident that the CS-ME allows for longer lifetime values for all devices, whereas the ESF causes device shutdowns in just 5 hours and the Re-OPSEC in 3 hours. It results in an extension of around 40% in the lifetime of the first device whose battery runs out with respect to ESF, and an extension of around 130% with respect to Re-OPSEC, mainly due to the fact that using Re-OPSEC tasks are reassigned at each time period, causing a conspicuous overhead. It is important to note that ideally, using the CS-ME strategy, the devices should have shut down (almost) together, but it did not happen due to the fact that a real system has random components, such as smartphone usage, signal strength, network congestion or latency in acquiring the position, which are not predictable. Those aspects introduce an error on the estimated battery drain which accumulates during the day and affects the lifetime of the device. To reduce the error between the predicted and observed lifetime, the system should know every change on each node. Of course, this information requires energy in order to be sent to the SVOs and, consequently, it introduces a global overhead resulting in a reduction of the total lifetime. Therefore, there is a trade-off between the error on predicted and effective lifetime.

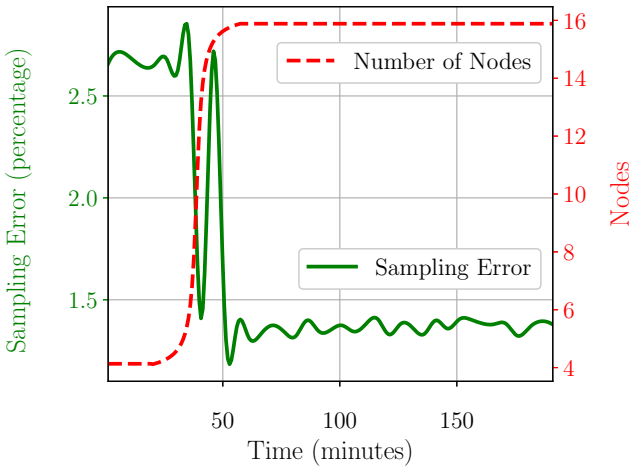


Fig. 8: Relative error on the resulting sampling frequency and participating number of devices in the geofence over the time

Figure 8 shows the relative error of the sampling rate in percentage, which is measured as the difference in the number of received samples per minute with respect to the target one. In the same Figure, we are measuring the number of sensors that were available in the considered geofence. Note that each time a new device enters, the allocation algorithm is run again by the CS-ME and the new sampling frequencies are assigned to the devices. Whereas the error was always below 3%, it quickly decreased when the number of devices grew. This phenomenon is due to the fact that the frequency assigned to each device decreased so that the impact of the latency in the sampling and transmission operation in the physical device was reduced.

To compare the adopted MSC policy with others, we have considered the Collector Friendly (CFP) and the Smartphone Friendly (SFP) policies implemented in an MSC framework that relies on distributed algorithms where each device is capable to locally compute all parameters required to take local sensing decisions. The CFP is intended to make the smartphone friendlier with the nodes community, causing the device to spend more energy to improve data redundancy and reliability. The SFP has been designed to avoid executing unnecessary sensing tasks and energy is consumed only if strictly necessary (the nodes a more selfish). These represent two opposite polities, whereas the one we propose sits in the middle. Indeed, the policy adopted by the proposed CS-ME is a trade-off between the other two policies because the workload is distributed among all the nodes with a logic that is based on the global knowledge of all nodes status (battery level, battery drain, connection technology, among others),

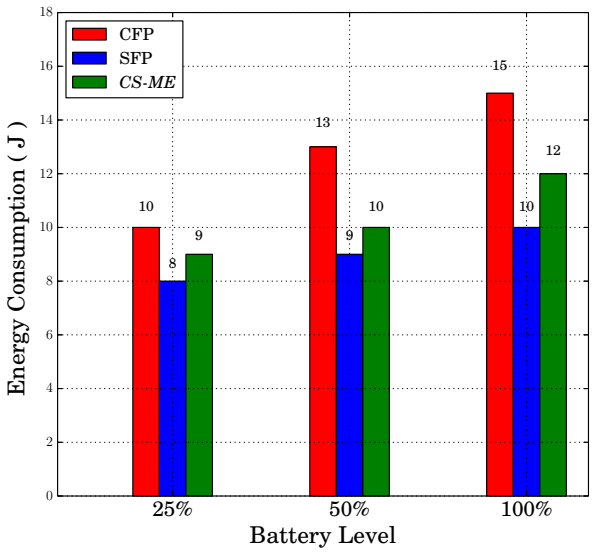


Fig. 9: Comparison between different collecting polices, i.e. *Collector Friendly Policy* (CFP), *Smartphone Friendly Policy* (SFP) adopted in [33] and the energy-aware policy adopted by our proposed CS-ME in terms of energy consumption

TABLE VI: Impact of the Minimum Sampling Frequency on battery lifetime

Frequency $F^{MSF}(Hz)$	Mean Lifetime τ (HH:MM)
0.130	6:58
0.067	7:40
0.033	8:09

taking into account the real capabilities of each device. Figure 9 compares the energy consumption of three users for these different collecting data policies, considering $F_k^{MSF} = 0.4$ Hz during 10 minutes of simulation. Each user started from different battery levels, i.e., 25%, 50% and 75%. It can be noted as the proposed one stays in the middle. Whereas the proposed algorithm consumes slightly more than the SFP, it still guarantees that the necessary amount of sensed data has been delivered, which is not the case for the SFP. On the other hand, the CFP may introduce unnecessary sensing tasks when not needed and also not fulfilling the F_k^{MSF} requirement.

In Figure 10 we show the total amount of data generated using the three polices during the 10-minute session; the expected amount of data is also shown on the basis of the required $F_k^{MSF} = 0.4$ Hz. The produced information is encoded in JSON format and transmitted as a string. The amount of data generated for the temperature is higher than that of the pressure because it has one more character on average to be transmitted. It can be noted that the proposed algorithm is much closer to the target rate with respect to the alternative approaches and generates always a rate of samples which is fulfilling the application requirements.

We also investigated the impact of frequency F^{MSF} and node speed on network lifetime. As expected the lower the minimum sampling frequency the higher the lifetime of nodes

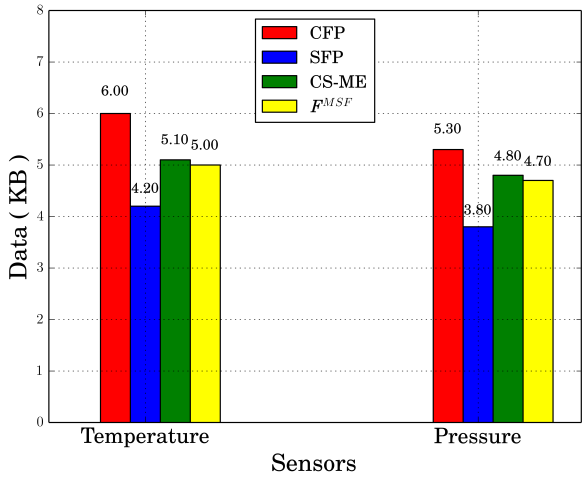


Fig. 10: Comparison between different collecting policies, i.e. *Collector Friendly Policy* (CFP), *Smartphone Friendly Policy* (SFP) adopted in [33] and the energy-aware policy adopted by our proposed CS-ME in terms of data generated during the 10-minute session with $F_k^{MSF} = 0.4$ Hz

TABLE VII: Impact of the nodes speed on battery lifetime

Speed (Km/h)	Mean Lifetime τ (HH:MM)
5	7:40
10	6:29
15	5:50

(see Table VI); however, we observed that a reduction of F^{MSF} of 75% has brought to a quite lower increase in the lifetime (around 15%). This is due to the fact that the MCS tasks are only partially affecting the energy consumption in the involved nodes. We also observed that the higher the average speed of the nodes the lower the lifetime (see Table VII). The reason is that in this case, it happens that the nodes need to be reconfigured more frequently due to a higher number of nodes leaving the entering the reference geofence.

In order to quantitatively assess how the mobility affects the lifetime of the network, we have simulated a scenario by varying the speed of the nodes so as to analyze the impact of two important parameters in our algorithm, i.e., α and β, which impact on the thresholds for accepting a node in the MCS assignment on the basis of its residual energy and speed, respectively (see equations 1 and 2). Figure 11 shows the network lifetime with respect to α and β when varying in a range of values for which the impact on the lifetime is maximum. From this, we can make two important observations. The first is related to the fact that increasing β we see an improvement in the lifetime as more nodes are involved in the sensing activity even if at increasing speed. However, this benefit soon disappears due to the fact that it is compensated by an increase in the consumption of the energy due to the management of frequent entrance and exit of nodes in the process, which entails for an increase in the energy wasted for management updates rather than for useful crowdsensing activities. The second observation is related to the area with high values of

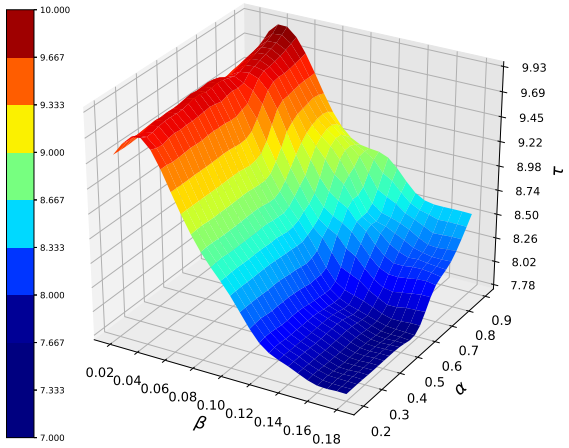
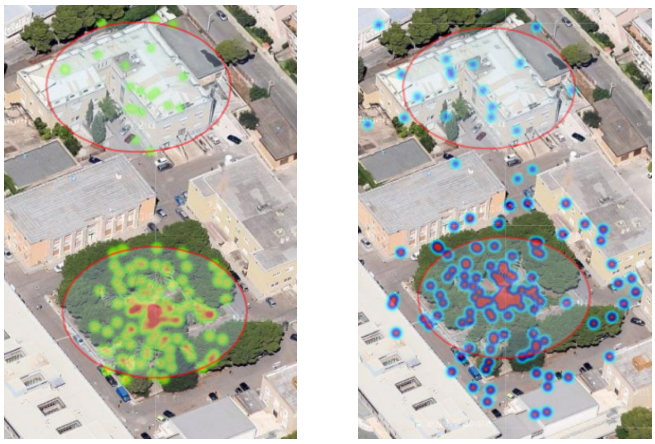


Fig. 11: Impact of α and β on network lifetime, expressed in hours



(a) Case 1 $\beta = 0.03$ (b) case 2 $\beta = 0.15$

Fig. 12: Impact of β on the number of samples collected outside the geofence

β: here we see that increasing α leads an improvement in the lifetime because the accepted nodes have high residual energy, so they can contribute more to reach the F_k^{MSF}. Keeping β high but decreasing α means accepting high-speed nodes even if they have low residual energy, so their assigned sampling frequencies $f_{i,k}$ are low. The other nodes have spent energy to reconfigure the network, but the accepted nodes can't contribute enough to sampling activities in order to increase the lifetime, because the Geofence's crossing time is too short. As already specified in the Introduction, our aim is not to propose a new crowdsensing application but rather a new specific functionality that several crowdsensing applications can rely on. However, to qualitatively show how the proposed system behavior may impact on the performance of a possible crowdsensing application, we have considered the monitoring of the local temperature per geofence. We have then analyzed how the proposed algorithm can be accurate in collecting data from the only devices that belong to the inquired geographical area. In particular, we focused on two cases with different values of the parameter β: in case 1 $\beta = 0.03$ and in case 2 $\beta = 0.15$ (α has been kept equal to 0.8). From Figure 12, it can be noted that the number of measurements taken from outside the geofence is higher in case 2 with respect to case 1 as faster nodes have been taken; indeed, high values of β have a negative impact on both nodes lifetime and precision in the identification of nodes in the geofence.

VI. Conclusions and Future Works

This paper addresses the issue of resource management in geolocated sensing applications, proposing a solution that relies on the SIoT Lysis platform. More specifically, we focused our study to MCS applications. The Social IoT model is exploited to find the potential members of the crowdsensing population and to make the nodes exchange the energy consumption profile among nodes with similar characteristics. From the emulation-based experiments, we observed the following results. We have been able to extend the time needed to completely deplete the battery of the first device of around 40% with respect to the case in which the sampling frequency of all the nodes is kept equal and constant. Additionally, due to the latency in performing the sampling and transmission observed in real devices, we experienced a sampling frequency error of around 2%, which decreases as the number of devices increases. Future developments will primarily focus on extending the resource allocation algorithm to other types of resources.

Future developments will primarily focus on extending the resource allocation algorithm to other types of resources. Indeed, the system can be extended to actuators as well. In this case, the energy models should be deeply modified so as to account for the energy consumption due to acting devices. Furthermore, more heterogeneous devices such as vehicles, bicycles will be considered as participants with different needs, resources and capabilities. Finally, a deeper analysis will be performed over the different relation types so as to develop an even more realistic emulator of social devices.

References

[1] G. Fortino and P. Trunfio, *Internet of things based on smart objects: Technology, middleware and applications*. Springer, 2014.

[2] F. Al-Turjman, A. Betin-Can, E. Ever, and S. Alturjman, "Ubiquitous cloud-based monitoring via a mobile app in smartphones: An overview," in *Smart Cloud (SmartCloud), IEEE International Conference on*. IEEE, 2016, pp. 196–201.

[3] B. Guo, Z. Wang, Z. Yu, Y. Wang, N. Y. Yen, R. Huang, and X. Zhou, "Mobile crowd sensing and computing: The review of an emerging human-powered sensing paradigm," *ACM Computing Surveys (CSUR)*, vol. 48, no. 1, p. 7, 2015.

[4] 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.

[5] G. Fortino, A. Guerrieri, W. Russo, and C. Savaglio, "Integration of agent-based and cloud computing for the smart objects-oriented iot," in *Computer Supported Cooperative Work in Design (CSCWD), Proceedings of the 2014 IEEE 18th International Conference on*. IEEE, 2014, pp. 493–498.

[6] G. Fortino, W. Russo, C. Savaglio, W. Shen, and M. Zhou, "Agent-oriented cooperative smart objects: From iot system design to implementation," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, no. 99, pp. 1–18, 2017.

[7] J. Yang, H. Zhang, Y. Ling, C. Pan, and W. Sun, "Task allocation for wireless sensor network using modified binary particle swarm optimization," *Sensors Journal, IEEE*, vol. 14, 2014.

[8] V. Pilloni, E. Abd-Elrahman, M. Hadji, L. Atzori, and H. Afifi, "Iot_prose: Exploiting 3gpp services for task allocation in the internet of things," *Ad Hoc Networks*, vol. 66, 2017.

[9] C. Perera, D. S. Talagala, C. H. Liu, and J. C. Estrella, "Energy-efficient location and activity-aware on-demand mobile distributed sensing platform for sensing as a service in iot clouds," *IEEE Transactions on Computational Social Systems*, vol. 2, no. 4, pp. 171–181, 2015.

[10] L. Skorin-Kapov, K. Pripužić, M. Marjanović, A. Antonić, and I. P. Žarko, "Energy efficient and quality-driven continuous sensor management for mobile iot applications," in *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2014 International Conference on*. IEEE, 2014, pp. 397–406.

[11] S. Bradai, S. Khemakhem, and M. Jmaiel, "Re-opsec: Real time opportunistic scheduler framework for energy aware mobile crowdsensing," in *Software, Telecommunications and Computer Networks (SoftCOM), 2016 24th International Conference on*. IEEE, 2016, pp. 1–5.

[12] 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.

[13] H. Ma, D. Zhao, and P. Yuan, "Opportunities in mobile crowd sensing," *IEEE Communications Magazine*, vol. 52, no. 8, pp. 29–35, 2014.

[14] J. Soldatos, N. Kefalakis, M. Hauswirth, M. Serrano, J.-P. Calbimonte, M. Riahi, K. Aberer, P. P. Jayaraman, A. Zaslavsky, I. P. Žarko *et al.*, "Openiot: Open source internet-of-things in the cloud," in *Interoperability and Open-Source Solutions for the Internet of Things*. Springer, 2015, pp. 13–25.

[15] F. Al-Turjman, Y. K. Ever, E. Ever, H. X. Nguyen, and D. B. David, "Seamless key agreement framework for mobile-sink in iot based cloud-centric secured public safety sensor networks," *IEEE Access*, vol. 5, pp. 24 617–24 631, 2017.

[16] A. M. Ortiz, D. Hussein, S. Park, S. N. Han, and N. Crespi, "The cluster between internet of things and social networks: Review and research challenges," *Internet of Things Journal, IEEE*, vol. 1, no. 3, pp. 206–215, 2014.

[17] L. Atzori, A. Iera, G. Morabito, and M. Nitti, "The social internet of things (siot)–when social networks meet the internet of things: Concept, architecture and network characterization," *Computer Networks*, vol. 56, no. 16, pp. 3594–3608, 2012.

[18] F. Al-Turjman, "5g-enabled devices and smart-spaces in social-iot: An overview," *Future Generation Computer Systems*, 2017.

[19] L. Ding, P. Shi, and B. Liu, "The clustering of internet, internet of things and social network," in *2010 Third International Symposium on Knowledge Acquisition and Modeling*, 2010, pp. 417–420.

[20] L. Militano, A. Orsino, G. Araniti, M. Nitti, L. Atzori, and A. Iera, "Trust-based and social-aware coalition formation game for multihop data uploading in 5g systems," *Computer Networks*, vol. 111, pp. 141–151, 2016.

[21] R. Girau, S. Martis, and L. Atzori, "Lysis: a platform for iot distributed applications over socially connected objects," *IEEE Internet of Things Journal*, vol. PP, no. 99, pp. 1–1, 2016.

[22] M. Nitti, V. Pilloni, G. Colistra, and L. Atzori, "The virtual object as a major element of the internet of things: a survey," *IEEE Communications Surveys Tutorials*, vol. PP, no. 99, pp. 1–1, 2015.

[23] M. Zhang, P. Yang, C. Tian, S. Tang, X. Gao, B. Wang, and F. Xiao, "Quality-aware sensing coverage in budget-constrained mobile crowdsensing networks," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 9, pp. 7698–7707, 2016.

[24] S. Delpriori, V. Freschi, E. Lattanzi, and A. Bogliolo, "Efficient algorithms for accuracy improvement in mobile crowdsensing vehicular applications," in *UBICOMM 2015*, 2015, p. 158.

[25] L. Ruge, B. Altakrouri, and A. Schrader, "Soundofthecity-continuous noise monitoring for a healthy city," in *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2013 IEEE International Conference on*. IEEE, 2013, pp. 670–675.

[26] B. Predić, Z. Yan, J. Eberle, D. Stojanovic, and K. Aberer, "Exposuresense: Integrating daily activities with air quality using mobile participatory sensing," in *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2013 IEEE International Conference on*. IEEE, 2013, pp. 303–305.

[27] N. Lathia, V. Pejovic, K. K. Rachuri, C. Mascolo, M. Musolesi, and P. J. Rentfrow, "Smartphones for large-scale behavior change interventions." *IEEE Pervasive Computing*, vol. 12, no. 3, pp. 66–73, 2013.

[28] J. Goldman, K. Shilton, J. Burke, D. Estrin, M. Hansen, N. Ramanathan, S. Reddy, V. Samanta, M. Srivastava, and R. West, "Participatory sensing: A citizen-powered approach to illuminating the patterns that shape our world," *Foresight & Governance Project, White Paper*, pp. 1–15, 2009.

[29] Z. Wang, D. Zhang, X. Zhou, D. Yang, Z. Yu, and Z. Yu, "Discovering and profiling overlapping communities in location-based social networks," *IEEE Transactions on*

Systems, Man, and Cybernetics: Systems, vol. 44, no. 4, pp. 499–509, 2014.

[30] S. B. Eisenman, E. Miluzzo, N. D. Lane, R. A. Peterson, G.-S. Ahn, and A. T. Campbell, "Bikenet: A mobile sensing system for cyclist experience mapping," *ACM Transactions on Sensor Networks (TOSN)*, vol. 6, no. 1, p. 6, 2009.

[31] M. Z. Hasan and F. Al-Turjman, "Optimizing multipath routing with guaranteed fault tolerance in internet of things," *IEEE Sensors Journal*, vol. 17, no. 19, pp. 6463–6473, 2017.

[32] Ö. Yürür, C. H. Liu, Z. Sheng, V. C. Leung, W. Moreno, and K. K. Leung, "Context-awareness for mobile sensing: A survey and future directions," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 68–93, 2016.

[33] A. Capponi, C. Fiandrino, D. Kliazovich, P. Bouvry, and S. Giordano, "A cost-effective distributed framework for data collection in cloud-based mobile crowd sensing architectures," *IEEE Transactions on Sustainable Computing*, vol. 2, no. 1, pp. 3–16, Jan 2017.

[34] V. Pilloni, L. Atzori, and M. Mallus, "Dynamic involvement of real world objects in the iot: a consensus-based cooperation approach," *Sensors*, vol. 17, no. 3, p. 484, 2017.

[35] Y. Yun, Y. Xia, B. Behdani, and J. C. Smith, "Distributed algorithm for lifetime maximization in a delay-tolerant wireless sensor network with a mobile sink," *Mobile Computing, IEEE Transactions on*, vol. 12, no. 10, pp. 1920–1930, 2013.

[36] M.-R. Ra, B. Priyantha, A. Kansal, and J. Liu, "Improving energy efficiency of personal sensing applications with heterogeneous multi-processors," in *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, ser. UbiComp '12. New York, NY, USA: ACM, 2012, pp. 1–10. [Online]. Available: http://doi.acm.org/10.1145/2370216.2370218

[37] "3gpp ts 25.331, radio resource control (rrc); protocol specification, may 1999."

[38] "3gpp ts 36.331, e-utra; radio resource control (rrc) protocol specification, may 2008."

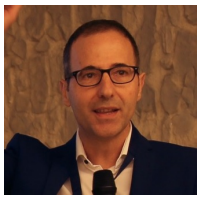
[39] S. Tarkoma, M. Siekkinen, E. Lagerspetz, and Y. Xiao, *Smartphone energy consumption: modeling and optimization*. Cambridge University Press, 2014.

[40] M. A. Hoque, M. Siekkinen, K. N. Khan, Y. Xiao, and S. Tarkoma, "Modeling, profiling, and debugging the energy consumption of mobile devices," *ACM Computing Surveys (CSUR)*, vol. 48, no. 3, p. 39, 2016.

[41] A. Rahmati and L. Zhong, "Human–battery interaction on mobile phones," *Pervasive and Mobile Computing*, vol. 5, no. 5, pp. 465–477, 2009.

[42] N. Vallina-Rodriguez, P. Hui, J. Crowcroft, and A. Rice, "Exhausting battery statistics: understanding the energy demands on mobile handsets," in *Proceedings of the second ACM SIGCOMM workshop on Networking, systems, and applications on mobile handhelds*. ACM, 2010, pp. 9–14.

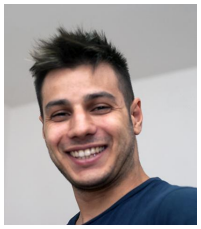
[43] G. Fortino, R. Gravina, W. Russo, and C. Savaglio, "Modeling and simulating internet-of-things systems: a hybrid agent-oriented approach," *Computing in Science & Engineering*, vol. 19, no. 5, pp. 68–76, 2017.



Luigi Atzori (SM'09) is Associate Professor at the Department of Electrical and Electronic Engineering at the University of Cagliari (Italy), where he leads the laboratory of Multimedia and Communications. His interests are in: multimedia communications, NGN service management, and IoT. He is the coordinator of the Marie Curie Initial Training Network on QoE for multimedia services (qoenet-itn.eu), which involves ten European Institutions in Europe and one in South Korea. He is member of the steering committee for the IEEE Trans. on Multimedia, member of the editorial board of the IEEE IoT, the Elsevier Ad Hoc Networks and the Elsevier Digital Communications and Networks journals.

Roberto Girau is Assistant Professor at the University of Cagliari, Italy. Dr. Girau received his Ph.D. degree in electronic engineering and computer science from the University of Cagliari in 2016. His main research interests are on Cloud Networking and Internet of Things (IoT), particularly on the creation of a network infrastructure to allow the objects to organize themselves according to a social structure.

Virginia Pilloni (S'11, M'14) is an Assistant Professor at the University of Cagliari. She was awarded with the Master Degree in Telecommunication Engineering with full marks in 2009 at the University of Cagliari. From November 2011 to April 2012 she was a visiting PhD student at the CCSR at the University of Surrey. She has been involved in several research projects, among which ACCUS and DEMANES (funded by FP7, Artemis-JU). In 2013 she was awarded with the PhD degree with Doctor Europaeus mention. Her main research interests are Internet of Things and Wireless ad-hoc networks, with particular attention to the improvement of their performance through task allocation.



Marco Uras is an Assistant Researcher at the University of Cagliari. He received his BSc. in Electrical and Electronic Engineering at the University of Cagliari in September 2016, defending a thesis entitled 'Development of energy-aware and location-aware algorithms for Mobile Crowdsensing'. He is currently enrolled in the master's degree program of Telecommunication Engineering at University of Cagliari. Since November 2016 he has been working for the Department of Electrical and Electronic Engineering of the University of Cagliari. In the same year he started a collaboration with MCLab research group, in the area of IoT and Mobile Crowdsensing.